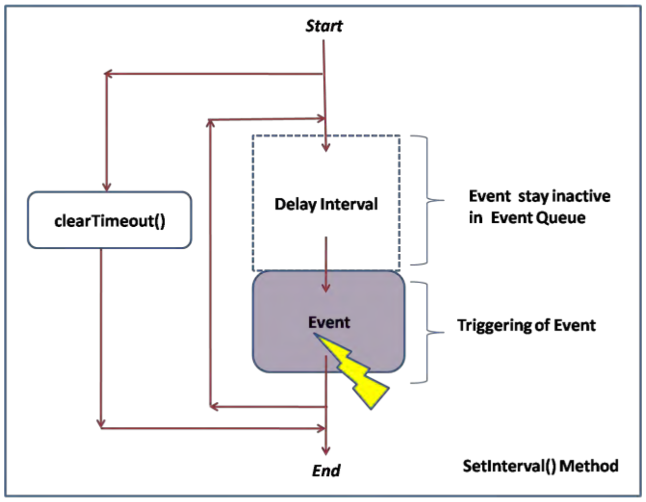
MCA-304

Advance Web Technology



VARDHMAN MAHAVEER OPEN UNIVERSITY

KOTA

www.vmou.ac.in

MCA-304

Vardhman Mahaveer Open University, Kota

Advance Web Technology

Course Development Committee

Chair Person

Prof. Ashok Sharma
Vice-Chancellor
Vardhman Mahaveer Open University, Kota

Prof. L.R. Gurjar
Director Academic
Vardhman Mahaveer Open University, Kota

Convener and Members

Convener
Neeraj Arora
Assistant Professor, Computer Science
School of Science and Technology,
Vardhman Mahaveer Open University, Kota.

Members

1. **Prof. (Dr.) Reena Dadich**
   Professor and Head (CS)
   University of Kota, Kota
2. **Prof. (Dr.) N.K. Joshi**
   Professor (CS) and Director, MIMT, Kota
3. **Dr. Harish Sharma**
   Associate Professor, CSE Deptt.
   Rajasthan Technical University, Kota
4. **Mr. Abhishek Nagar,**
   Programmer Officer, VMOU, Kota
5. **Dr. Anuradha Dubey**
   Deputy Director
   School of Science &Technology
   Vardhman Mahaveer Open University, Kota

Editor & Unit Writers **MCA-304: Advance Web Technology**

Editor

Neeraj Arora

Assistant Professor and Convener, Computer Science, School of Science and Technology, Vardhman Mahaveer Open University, Kota.

Unit Writers	*Units*
Prof. Reena Dadich Professor and Head, Department of CS & Informatics, University of Kota, Kota.	**1, 9**
Mrs. Jyoti Lakhani Asstt. Prof. & HOD, Computer Science, Maharaja Ganga Singh University, Bikaner	**2, 3**
Mr. Sunil Sharma Lecturer, Computer Science, Govt. Polytechnic College, Jalawar.	**4**
Mr. Neeraj Arora Assistant Professor, Computer Science, Vardhman Mahaveer Open University, Kota	**6, 8, 10, 11**
Mr. Kamal Kulshrestha Associate Prof. & HOD, Computer Applications, MIMT, Kota	**7**
Mr. Om Prakash Suthar Asst. Prof. (SR. Scale), Department of CSE, J.I.E.T., Jodhpur.	**12**
Mr. Parth Vidhayarthi Asst. Professor (Computer Science & Engg. Deptt),Career Point University, Kota	**5**
Mr. Sanjay Kumar Anand Assistant Professor, Department of Computer Science, Central University of Rajasthan, Kishangarh, Ajmer	**13**

Academic and Administrative Management

Prof. Ashok Sharma
Vice-Chancellor, V.M.O. University, Kota

Prof. L.R. Gurjar
Director (Academic), V.M.O. University, Kota

Dr. Shiv Kumar Mishra
Director (MP&D), V.M.O. University, Kota

Print: 2018 **ISBN: : 978-81-8496-643-5**

All Right reserved. No part of this Book may be reproduced in any form by mimeograph or any other means without permission in writing from Vardhman Mahaveer Open University, Kota.

Printed and Published on behalf of the Registrar, Vardhman Mahaveer Open University, Kota.

Printed by :

MCA-304

Vardhman Mahaveer Open University, Kota

Advance Web Technology

Contents

Preface

The present book entitled "Advance Web Technology" has been designed so as to cover the unit-wise syllabus of MCA-304 course for MCA 3rd Year students of Vardhman Mahaveer Open University, Kota.

The book is dedicated to the description of the latest trends and happening in Web application development. The book starts with some introduction of Web Technologies like HTML, JavaScript and CSS. Some advance topics like AJAX, jQuery, cloud computing are also discussed in this book. For server side scripting language PHP is chosen with MySQL which are Open source software. The book ends with Web content management like Wordpress, Drupal and Joomla which can be used for creating a blog.

Each unit begins with objectives, introduction and principles together with illustrative and other descriptive material .The illustrative examples serve to illustrate and amplify the theory, bring into focus on important concepts. The units have been written by various experts in the field. We believe that this book is well suited to self-learning. The text is written in a logical sequence and is beneficial for students. The concise and sequential nature of the book makes it easier to learn. Although we have made all efforts to make the text error free, yet errors may remain in the text. We shall be thankful to the students and teachers alike if they point these out to us. Any further comments and suggestions for future improvement are welcome and will be most gratefully acknowledged.

UNIT-1
Introduction to Web Technology

Structure of the Unit

1.0 Objective

This chapter provides a general overview of

- HTML
- Document Object Model (DOM)
- JavaScript
- HTML DOM and JavaScript

1.1 Introduction

A Web is a universal medium for exchanging data, information, communication, and conducting business. It provides services that can be accessed from any computer/device connected to the Internet. Organizations and individuals can provide information and services to an international audience by setting up web sites on the Internet. Today, millions of web sites provide services that facilitate communication, share resources, entertain, and conduct commercial business.
The information on the WWW is stored in Web documents residing on servers. A Web document, or Web page, is an electronic document that stores a set of related information. Technologies that are used to create such documents are known as Web Technologies. Now-a-days, a number of tools and technologies are available for creating web sites. In this chapter we will be learning one of these technologies known as HTML. We will also learn the use of Java Script and Document Object Model.

HTML is one of the oldest yet the most popular and easy way of creating web content. Web pages are generally written in HTML. As a result, Web pages are also referred to as HTML documents.

HTML stands for **H**ypertext **M**arkup **L**anguage, and it is the most widely used language to create Web Pages.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- HTML is a **Markup Language** used to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

HTML is a descriptive language that helps convert ordinary text into hypertext by adding special code called **tags** or **elements**. An HTML tag consists of a left angular bracket (<), the name of the tag, and a right angular bracket (>). HTML tags are not case sensitive. Tags tell an HTML interpreter (a web browser) how to display the HTML document. When we enclose text within tags, it becomes an ***element***. An element is used to describe the contents in an HTML document. Elements are the main markup components of HTML. Each element represents a piece of data that is identified by a tag.

A tag may also include ***attributes***, which provide additional information about an element. An element can have one or more attributes. Some attributes may be mandatory while others may be optional.

1.2 HTML Document Structure

A typical HTML document will have the following structure:

```
<html>
<head> Document header related tags </head>
<body> Document body related tags </body>
</html>
```

Each HTML page starts with <html> tag and ends with </html> tag. Every HTML document has two sections – **header** and **body** contained within the HTML tags. The header section begins with <head> tag and ends with </head> tag. It contains information about the document, which is not visible to the user. The header section compulsorily includes a <title> tag that contains the title of the page. This title appears in the title bar of the browser. The other components in header section are Prologue, Base, Meta, Script, Link and Style. The body section begins with <body> tag and ends with </body> tag. The body section contains the actual contents of the page. The comments are enclosed in the tags <!—comment -->. All HTML files have **.htm** or **.html** file extensions.

Basic HTML Document

In its simplest form, following is an example of an HTML document:

```
<html>
<head> <!—header section -->
<title> My First Web Page</title>
</head>
<body><!—body section -->
 This is my first Web page created using HTML.
</body>
</html>
```

1.2.1 HEAD SECTION

Following table illustrates the tags of HEAD Element:

Tag Name	Description
Prologue	This component is only a comment used to specify the version of HTML. <! DOCTYPE HTML 5.0>
Link	This tag is used to inform the browser's previous and next document, to link the banner and to inform about the location of the base document. This tag is also used to specify relationships between the current document and external resource. Like external

	style sheet file. <Link rel = previous href=”prev.html”> <Link rel = next href=”next.html”> <Link rel = banner href=”banner.gif”> <link rel="stylesheet" type="text/css" href="style.css">
Base	The <base> tag declares the global reference values for **href** and **target** attributes. The attribute *href* specifies the base URL for all relative URLs in a page, which means all the other URLs will be concatenated into base URL while locating for the given item. <base href="base_url"> The attribute *target* specifies the default frame name for all links. <base target = “frame_name”>
Meta	The HTML <meta> tag is used to provide metadata about the HTML document which includes information about page expiry, page author, list of keywords, page description etc. <!-- Provide list of keywords --> <meta name="keywords" content="C, C++, Java, PHP, Perl, Python"> <!-- Provide description of the page --> <meta name="description" content="Simply Easy Learning by Tutorials Point"> <!-- Author information --> <meta name="author" content="xyz"> <!—Page content type --> <meta http-equiv="content-type" content="text/html; charset=UTF-8"> <!-- Page refreshing delay --> <meta http-equiv="refresh" content="30"> <!-- Page expiry --> <meta http-equiv="expires" content="Sat, 31 Dec 2016 14:25:27 GMT">

Script	The HTML <script> tag is used to include either external script file or to define internal script for the HTML document. <script language="JavaScript" type="text/JavaScript"> ----- script code </script>
Style	The HTML <style> tag is used to specify style sheet for the current HTML document. <style type="text/css"> -----style statements </style>
Title	The HTML <title> tag is used for specifying the title of the HTML document. <title>HTML Title Tag Example</title>

1.2.2 BODY SECTION

The actual contents of the web page are placed in the body section. The <body> tag consists of various attributes as listed below.

● **bgcolor**	:	specifies the background color of the web page.
● **Link**	:	specifies the color of the unvisited link.
● **Alink**	:	specifies the color of the active link.
● **Vlink**	:	specifies the color of the visited link.
● **Text**	:	specifies the text color.

Example,

```
<body bgcolor="green" link="blue" alink="purple" vlink="pink" text="white">
```

The document's contents can be placed between various tags for special appearance. Main tags of the BODY element are as follows:-

Tag Name	**Attribute Name**	**Attribute Function**
 		Inserts a line break.
<p> … </p>		Begins and ends a paragraph.

<h1>…</h1> to <h6>…</h6>	align	Specifies text as heading. The **align** [left, center, right] attribute is used to align the header text as left-justified or centered or as right-justified.
<font>	Color	Specifies the color of the text.
	Face	Specifies the typeface of the text.
	Size	Specifies the size of the text.
<center>		Places any content in the center of the page or any table cell.
<hr>		Inserts a shaded line, also called a horizontal rule.
Text Formatting tags		
<b> </b> <strong> </strong>		Makes the text bold
<i> </i> <em> </em>		Makes the text in italics
<u>...</u>		Makes the text underlined
<strike>...</strike>		Displays a thin line through the text
^{...}		Text is written in superscript
_{...}		Text is written in subscript
<img>	Src	To specify the location of the image file to be used.
	Height & Width	To Specify height & width of the image in terms of either pixels or percentage of its actual size.
	Border	To specify border thickness in terms of pixels
	hspace & vspace	Specifies horizontal & vertical margin of image.
	Align	Sets the alignment of image. Values can be left, center or right.

	Alt	Alternate text to be displayed in place of image.

1.3 HTML Lists

HTML provides three ways for specifying lists of information. All lists must contain one or more list elements. Lists are of three kinds:

- **<ul>** - An unordered list. This will list items using plain bullets.
- **<ol>** - An ordered list. This will use different schemes of numbers to list our items.
- **<dl>** - A definition list. This arranges our items in the same way as they are arranged in a dictionary.

- **HTML Unordered Lists**

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML **<ul>** tag. Each item in the list is marked with a bullet. To specify items of the list <li> tag is used. We can use **type** attribute for <ul> tag to specify the type of bullet. By default, it is a disc. Following are the possible options:

```
<ul type="square">
<ul type="disc">
<ul type="circle">
```

- **HTML Ordered Lists**

If we wish to put items in a numbered list instead of bulleted, then HTML ordered list will be used. This list is created by using **<ol>** tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>. We can use **type** attribute for **<ol>** tag to specify the type of numbering. By default, it is a number. The **start** attribute for <**ol**> tag is used to specify the starting point of numbering. Following are the possible options:

<ol type="1" start="4"> - Default-Case Numerals starts with 4.
<ol type="I" start="4"> - Upper-Case Numerals starts with IV.
<ol type="i" start="4"> - Lower-Case Numerals starts with iv.
<ol type="a" start="4"> - Lower-Case Letters starts with d.
<ol type="A" start="4"> - Upper-Case Letters starts with D.

- **HTML Definition Lists**

  HTML supports a list style which is called **definition list.** Each element in this list is labeled with a word instead of a bullet or number. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following tags.

- <dl> - Defines the start of the list
- <dt> - A term
- <dd> - Term definition
- </dl> - Defines the end of the list

1.4 HTML Tables

The HTML tables allow us to arrange data like text, images, links, other tables, etc. into rows and columns of cells. The HTML tables are created using the **<table>** tag in which the **<tr>** tag is used to create table rows and **<td>** tag is used to create data cells. Table heading can be defined using **<th>** tag. Following is the list of attributes related to HTML Tables :

Attributes	Description
Align	Sets the alignment of table. Values can be left, center or right.
Border	Specifies border thickness in terms of pixels.
Bgcolor	Sets background color for whole table or just for one cell.
Background	Sets background image for whole table or just for one cell.
Cellpadding	Sets the distance between cell borders and the content within a

	cell.
Cellspacing	Sets the width of the cell border.
Height/Width	To Specify height / width of the table in terms of pixels/percentage.
Attributes of <td> and <th> tag :	
Align	Sets the alignment of contents in a cell. Values can be left, center or right.
Colspan	Used to merge two or more columns into a single column.
Rowspan	Used to merge two or more rows.
Attributes of <tr> tag :	
Align	Sets the alignment (left, center, right) of contents in cells of a row.

1.5 Linking

A webpage can contain various links that take us directly to other pages and even specific parts of a given page. These links are known as hyperlinks. We can create hyperlinks using text or images available on a webpage.

1.5.1 External Linking

A link is specified using HTML tag **<a>**. This tag is called **anchor tag** and anything between the opening **<a>** tag and the closing **</a>** tag becomes part of the link and a user can click that part to reach to the linked document. This tag requires an attribute **href** to mark the location of the object to be linked.

HTML Email Tag

HTML **<a>** tag provides the option to specify an email address to send an email. While using <a> tag as an email tag, we will use **mailto: email address** along with ***href*** attribute.

Following is the syntax of using **mailto** instead of using http.

```
<a href= "mailto:abc@example.com">Send Email</a>
```

We can specify a default ***email subject*** and ***email body*** along with our email address.

Following is the example to use default subject and body.

```
<a href="mailto:abc@example.com?subject=Feedback&body=Message">
Send Feedback
</a>
```

1.5.2 Internal Linking (Linking to a Page Section)

We can create a link to a particular section of a given webpage by using **name** attribute. This is a two-step process.
First create a link to the place where we want to reach with-in a webpage and name it using **<a...>** tag as follows:

```
<h1>HTML Text Links <a name="top"></a></h1>
```

Second step is to create a hyperlink to link the document and place where we want to reach:

```
<a href="html_text_links.html#top">Go to the Top</a>
```

1.5.3 Image Links

We have seen how to create hyperlink using text, now we will learn how to use images to create hyperlinks.

Example

```
<p>Click following link</p>
<a href="http://www.Chapter.com" target="_self">
<img src="/images/logo.png" alt="Image Link" border="0">
</a>
```

1.6 HTML frames

HTML frames are used to divide the browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns. To use frames on a page we use **<frameset>** tag instead of **<body>** tag. The **<frameset>** tag defines, how to divide the window into frames. The **rows** attribute of **<frameset>** tag defines horizontal frames and **cols** attribute defines vertical frames. Each frame is indicated by **<frame>** tag and it defines which HTML document shall open into the frame.

Following are important attributes of the **<frameset>** tag:

Attribute	**Description**
cols	Specifies how many columns are contained in the frameset and the size of each column. We can specify the width of each column in one of the four ways: Absolute values in pixels. For example, to create three vertical frames, use *cols="100, 500,100"*. A percentage of the browser window. For example, to create three vertical frames, use *cols="10%, 80%,10%"*. Using a wildcard symbol. For example, to create three vertical frames, use *cols="10%, *,10%"*. In this case wildcard takes remainder of the window.
rows	This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset. For example, to create two horizontal frames, use *rows="10%, 90%"*. We can specify the height of each row in the same way as explained above for columns.
border	This attribute specifies the width of the border of each frame in pixels. For example, border="5". A value of zero means no border.

frameborder	This attribute specifies whether a three-dimensional border should be displayed between frames. This attribute takes value either 1 (yes) or 0 (no). For example frameborder="0" specifies no border.
framespacing	This attribute specifies the amount of space between frames in a frameset. This can take any integer value.

The <frame> Tag Attributes

Following are the important attributes of <frame> tag:

Attribute	Description
src	This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL.
name	This attribute allows us to give a name to a frame. It is used to indicate which frame a document should be loaded into. This is especially important when we want to create links in one frame that load pages into an another frame, in which case the second frame needs a name to identify itself as the target of the link.
frameborder	This attribute specifies whether or not the borders of that frame are shown; it overrides the value given in the frameborder attribute on the <frameset> tag if one is given, and this can take values either 1 (yes) or 0 (no).
Marginwidth / marginheight	This attribute allows us to specify the width / height of the space between the left and right of the frame's borders and the frame's content. The value is given in pixels. For example marginwidth="10".
noresize	By default, we can resize any frame by clicking and dragging on the borders of a frame. The noresize attribute prevents a user from being able to resize the frame. For example noresize="noresize".
scrolling	This attribute controls the appearance of the scrollbars that appear on the frame.

1.7 HTML Forms

All the tags discussed above do not allow user to give inputs or press buttons, rather, they only display formatted content on the page. The **<form>** tag is used to create interactive page to collect information from the user and send to web server for further processing.

Following is a list of the most frequently used **form** attributes:

Attribute	**Description**
action	Specifies the location to which the contents of the form are submitted.
method	Specifies the method by which the browser sends the data in the form fields to the server for processing. The type of method can be either GET or POST.
target	Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
Enctype	Specifies the content type for encoding the data specified in the form fields.
Name	Specifies the name of the form.

HTML Form Controls

There are different types of form controls that we can use to collect data using HTML form:

● Text Input Controls	● Checkboxes Controls
● Radio Box Controls	● Select Box Controls
● File Select boxes	● Hidden Controls
● Buttons Controls	● Submit and Reset Button

Text Input Controls

There are three types of text input used on forms:

- **Single-line text input controls -** This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.
- **Password input controls** - This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag.

Attributes for <**input**> tag.

Attribute	**Description**
type	Indicates the type of input control and for text input control it will be set to **text** and for password input control it will be set to **password.**
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
value	This can be used to provide an initial value inside the control.
size	Allows to specify the width of the text/password input control in terms of characters.
maxlength	Allows to specify the maximum number of characters a user can enter into the text/password box.

- **Multi-line text input controls -** This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Attributes for <textarea> tag.

Attribute	**Description**
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
rows	Indicates the number of rows of text area box.
cols	Indicates the number of columns of text area box

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **checkbox**.

Example

```
<input type="checkbox" name="maths" value="on"> Maths
<input type="checkbox" name="physics" value="on"> Physics
```

Attributes for **checkbox**:-

Attribute	**Description**
type	Creates a checkbox for selecting multiple options from a set of alternatives.
name	Used to give a name to the control.
value	The value that will be used if the checkbox is selected.
checked	Set to *checked* if we want to select it by default.

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to **radio**.

Example

```
<input type="radio" name="subject" value="maths"> Maths
<input type="radio" name="subject" value="physics"> Physics
```

Attributes of radio button:-

Attribute	**Description**
type	Create a radio button for selecting any one option for a set of alternatives.
name	Used to give a name to the control.
value	The value that will be used if the radio box is selected.
checked	Set to *checked* if we want to select it by default.

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

Example

```
<select name="dropdown">
<option value="Maths" selected>Maths</option>
<option value="Physics">Physics</option>
</select>
```

Attributes of <select> tag:

Attribute	**Description**
name	Used to give a name to the control.
size	Specifies the number of visible items.
multiple	If set to "multiple" then allows a user to select multiple items from the menu.
Attributes of <option> tag:	
value	The value that will be used if an option in the select box box is selected.
selected	Specifies the initially selected value when the page loads.
label	An alternative way of labeling options

File Upload Box

If we want to allow a user to upload a file to our web site, we will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to **file**.

Example

```
<input type="file" name="fileupload" accept="image/*" />
```

Attributes of file upload box:

Attribute	Description
name	Used to give a name to the control which is sent to the server to be recognized and get the value.
accept	Specifies the types of files that the server accepts.

Button Controls

There are various ways in HTML to create clickable buttons. We can also create a clickable button using <input> tag by setting its type attribute to **button**. The type attribute can take the following values:

Type	Description
submit	This creates a button that automatically submits a form.
reset	This creates a button that automatically resets form controls to their initial values.
button	This creates a button that is used to trigger a client-side script when the user clicks that button.
image	This creates a clickable button but we can use an image as background of the button.

```
<input type="submit" name="submit" value="Submit" />
<input type="reset" name="reset" value="Reset" />
<input type="button" name="ok" value="OK" />
<input type="image" name="imagebutton" src="/html/images/logo.png" />
```

1.8 Document Object Model

Document Object Model (DOM) is a platform-independent and language-independent interface that permits scripts to access and update the content, structure, and style of a document. DOM includes a model in which a standard set of objects representing HTML documents are combined together, and an interface for accessing and manipulating them.

DOM expresses the structure of an HTML document in terms of-

- Types of tags on the page
- Number of tags on the page
- Order of tags on the page
- Properties of tags
- Style of an element

DOM creates an object for each element on the page. These objects have child objects and defined properties. Such child objects can have further child objects and defined properties.

1.9 HTML – STYLE SHEET

Cascading Style Sheets (CSS) provide easy and effective alternatives to specify various attributes for the HTML tags. Using CSS, we can specify a number of style properties for a given HTML element. Each property has a name and a value, separated by a colon (:). Each property declaration is separated by a semi-colon (;).

Example

First let's consider an example of HTML document which makes use of <font> tag and associated attributes to specify text color and font size:

```
<html>
<head> <title>HTML CSS</title> </head>
<body>
<p><font color="green" size="5">Hello, World!</font></p>
</body>
</html>
```

We can re-write above example with the help of Style Sheet as follows:

```
<html>
<head> <title>HTML CSS</title> </head>
<body>
```

```
<p style="color:green;font-size:24px;">Hello, World!</p>
</body> </html>
```

We can use CSS in three ways in our HTML document:

- **External Style Sheet**: Define style sheet rules in a separate .css file and then include that file in our HTML document using HTML <link> tag.
- **Internal Style Sheet**: Define style sheet rules in header section of the HTML document using <style> tag.
- **Inline Style Sheet**: Define style sheet rules directly along-with the HTML elements using **style** attribute.

Let's see all the three cases one by one with the help of suitable examples.

1.9.1 External Style Sheet

If we need to use our style sheet to various pages, then it's always recommended to define a common style sheet in a separate file. A cascading style sheet file will have extension as **.css** and it will be included in HTML files using <link> tag.

We define a style sheet file **style.css** which has following rules: Here we defined three CSS rules which will be applicable to three different classes defined for the HTML tags.

```
.red {color: red;}
.thick {font-size:20px;}
.green {color:green;}
```

Now let's make use of the above external CSS file in our following HTML document:

```
<html>
<head> <title>HTML External CSS</title>
<link rel="stylesheet" type="text/css" href="/html/style.css"> </head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
```

```
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body> </html>
```

1.9.2 Internal Style Sheet

If we want to apply Style Sheet rules to a single document only, then we can include those rules in header section of the HTML document using <style> tag. Rules defined in internal style sheet overrides the rules defined in an external CSS file. Let's re-write above example once again, but here we will write style sheet rules in the same HTML document using <style> tag:

```
<html>
<head> <title>HTML Internal CSS</title>
<style type="text/css">
.red{ color: red; }
.thick{ font-size:20px; }
.green{color:green; }
</style> </head>
<body>
<p class="red">This is red</p>
<p class="thick">This is thick</p>
<p class="green">This is green</p>
<p class="thick green">This is thick and green</p>
</body></html>
```

1.9.3 Inline Style Sheet

We can apply style sheet rules directly to any HTML element using **style** attribute of the relevant tag. This should be done only when we are interested to make a particular change in any HTML element only. Rules defined in line with the element overrides the rules defined in an external CSS file as well as the rules defined in <**style**> element. Let's re-write above example once again, but here we will write style sheet rules along with the HTML elements using **style** attribute of those elements.

```
<body>
<p style="color:red;">This is red</p>
<p style="font-size:20px;">This is thick</p>
<p style="color:green;">This is green</p>
<p style="color:green;font-size:20px;">This is thick and green</p>
</body>
```

1.10 HTML and JavaScript

JavaScript is a dynamic, script-based, light-weight, interpreted computer programming language with object-oriented capabilities. JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

The ECMA-262 Specification defined a standard version of the core JavaScript language.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

JavaScript supports the development of both Client and Server components of web-based applications. On the Client-side, it can be used to write programs that are executed by a web browser within the context of a web page. On the Server-side, it can be used to write Web Server programs that can process information submitted by a web browser and then, update the browser's display accordingly.

Figure 1.1 provides an overview of how JavaScript supports both client and server web programming

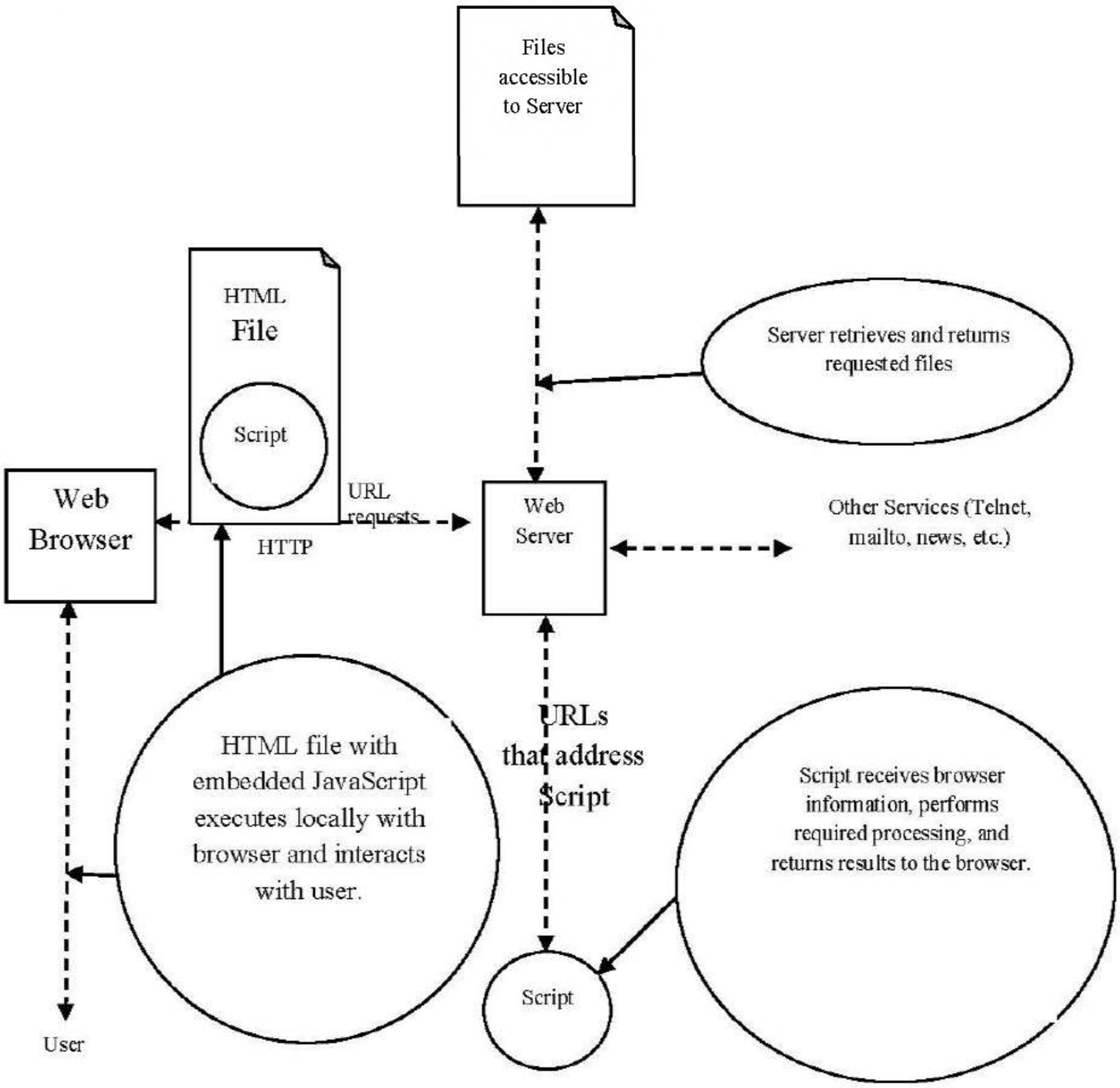

Files accessible to Server
HTML File
Script
Server retrieves and returns requested files
Web Browser
URL requests
HTTP
Web Server
Other Services (Telnet, mailto, news, etc.)
URLs that address Script
HTML file with embedded JavaScript executes locally with browser and interacts with user.
Script receives browser information, performs required processing, and returns results to the browser.
Script
User

Figure 1.1: JavaScript supports both Client and Server web applications

1.10.1 Embedding JavaScript in HTML:-

JavaScript statements can be included in HTML documents by enclosing the statements between an opening **<script>** tag and a closing **</script>** tag in a web page. Within opening **<script>** tag, the **language** attribute is set to **"JavaScript"** to identify the script as being JavaScript. The scrpt tag is used as follows:

```
<script language = "JavaScript">
 JavaScript statements
</script>
```

We can place the **<script>** tags, containing JavaScript statements, anywhere within the web page (**head** or **body** part), but it is normally recommended that we should keep it within the **<head>** tags to ensure that all JavaScript definitions have been made before the body of the document is displayed.

The script tag takes two important attributes:

- **Language:** This attribute specifies what scripting language you are using. Typically, its value will be "**JavaScript**". Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **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 our JavaScript syntax will look as follows.

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

Let us take a sample example to print out "Welcome to the world of JavaScript":

```
<html>
<head><title> Hello World!!! </title></head>
<body>
<script language="JavaScript">
 document.write ("Welcome to the world of JavaScript")
</script>
</body></html>
```

The script has a single statement calling a function document.write() which writes the supplied string into our HTML document.

HTML provides a method to conceal JavaScript code from browsers which do not support it. HTML comment tags (<!-- //-->) are used to surround JavaScript statements. The above code is rewritten using HTML Comment tags as follows :-

```
<html>
<head><title> Hello World!!! </title></head>
<body>
<script language="JavaScript">
<!-- Begin hiding JavaScript
 document.write ("Hello World!")
//  End hiding JavaScript -->
</script></body></html>
```

The **noscript** tag **:-**

The <noscript> tag is created for those browsers that can't or won't process JavaScript. We can add a noscript block immediately after the script block as follows:

```
<html>
 <body>
<script language="JavaScript" type="text/JavaScript">
<!--
  document.write ("Hello World!")
//-->
</script>
<noscript>
 Sorry...JavaScript is not enabled.
</noscript>
</body>
 </html>
```

Now, if the user's browser does not support JavaScript or JavaScript is not enabled, then the message from </noscript> will be displayed on the screen.

1.10.2 The SRC attribute of Script tag : JavaScript in External File :-

There are cases where we are reusing identical JavaScript code on multiple pages of a site. For such situations, the **<script>** tag provides a mechanism to allow us to store JavaScript in an external file and then include it into our HTML files.
To use JavaScript from an external file source, we need to write all our JavaScript source code in a simple text file with the extension ".js" and then include that file as shown below.

For example, if the file **src.js** contains the following code :

```
<!—
  document.write("This text is generated by code placed in src.js file.")
// -->
```

Then the HTML document would be written as below-

```
<html>
<head> <title> Using the SRC attribute of the Script tag </title></head>
<body>
<script language="Javascript" src="src.js" > </script>
</body> </html>
```

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. We can use spaces, tabs, and newlines freely in our program to format it.

Semicolons are Optional

Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows us to omit this semicolon if each of our statements is placed on a separate line. For example, the following code could be written without semicolons.

```
<script language="JavaScript" type="text/JavaScript">
<!--  var1 = 10
```

```
var2 = 20 //-->
</script>
```

But when formatted in a single line as follows, we must use semicolons:

```
<script language="JavaScript" type="text/JavaScript">
<! -- var1 = 10; var2 = 20;  //--> </script>
```

Case Sensitivity

JavaScript is a case-sensitive language. So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.

Comments in JavaScript

JavaScript supports both C-style and C++-style comments. Thus:

- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

The following example shows how to use comments in JavaScript.

```
<script language="JavaScript" type="text/JavaScript">    <!--
// This is a comment. It is similar to comments in C++
/*
* This is a multiline comment in JavaScript
* It is very similar to comments in C Programming
*/
//--> </script>
```

JavaScript Identifiers

An identifier is a name used to identify a variable, method, or an object. An identifier can be of two types – literals and variables. Literals have fixed values while variables have different values during execution.

JavaScript Data types

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 allows you to work with three primitive data types:

1. Numbers (Integers & Floating point), e.g., 123, 120.50 etc.
2. Strings of text, e.g. "This text string" etc.
3. Boolean, e.g. true or false.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**.

JavaScript Variables

Variables can be thought of as named containers. One can place data into these containers and then refer to the data simply by naming the container. Before using a variable in a JavaScript program, one must declare it. Variables are declared with the **var** keyword as follows.

```
<script language="JavaScript" type="text/JavaScript">
<!--
var money;
var name; //--> </script>
```

We can also declare multiple variables with the same **var** keyword as follows:

```
<script language="JavaScript" type="text/JavaScript">
<!--
var money, name;
//-->
```

```
</script>
```

Storing a value in a variable is called variable initialization. We can do variable initialization at the time of variable creation or at a later point in time when we need that variable. For instance, we might create a variable named money and assign the value 2000.50 to it later. For another variable, we can assign a value at the time of initialization as follows.

```
<script language="JavaScript" type="text/JavaScript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

JavaScript is **untyped l**anguage. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, we don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

JavaScript Variable Scope

The scope of a variable is the region of our program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables:** A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables:** A local variable will be visible only within a function where it is defined. Function parameters are always local to that function. Within the body of a function, a local variable takes precedence over a global variable with the same name. If we declare a local variable or

function parameter with the same name as a global variable, you effectively hide the global variable. Take a look into the following example.

```
<script language="JavaScript" type="text/JavaScript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
  var myVar = "local";  // Declare a local variable
  document.write(myVar);
} //--> </script>
```

Variable Names

While naming variables in JavaScript, we should keep the following rules in mind.

- Do not use any of the JavaScript reserved keywords as a variable name.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, **123test** is an invalid variable name but **_123test** is a valid one.
- JavaScript variable names are case-sensitive. For example, **Name** and **name** are two different variables.

JavaScript Expressions

An expression is a statement that evaluates to a value. The result can be of any type. Expressions contain variables, literals, operators and another expression. Some of the valid expressions are as follows:

Y=50;

Mg="Good Morning";

Res=x + y * t / d;

JavaScript Key / Reserved Words

A list of all the reserved words in JavaScript is given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	const	enum	function	interface	public	throw	volatile
boolean	continue	export	goto	long	return	throws	while
Break	debugger	extends	if	native	short	transient	with
Byte	default	false	implements	new	static	true	
Case	delete	final	import	null	super	try	
Catch	do	finally	in	package	switch	typeof	
Char	double	float	instanceof	private	synchronized	var	
Class	else	for	int	protected	this	void	

JavaScript Operators

Operators are commands that perform operations on variable and/or literals, and produce a result. The operators are of two types – unary and binary. Unary operators operate on only one operand and binary operators operate on two operands. Operators supported by JavaScript includes –

Operators	Syntax	Description / Result
1 - Assignment Operator :		
=	Res = x + y	Assigns the result of an expression to a variable.
2 - Compound assignment operators :		
+=	A += B	Its equivalent expression is A = A + B
-=	A -= B	Its equivalent expression is A = A - B
*=	A *= B	Its equivalent expression is A = A * B
/=	A /= B	Its equivalent expression is A = A / B
%=	A %= B	Its equivalent expression is A = A % B

3 - Arithmetic Operators :		
+	a + b	i. Returns sum of a and b if a and b are numbers. ii. Concatenates a and b if a and b are strings.
-	a – b	Returns difference of a and b
*	a * b	Returns product of a and b
/	a / b	Returns quotient (dividing a by b)
%	a % b	Returns remainder (dividing a by b)
++	++a or a++	Increment a by 1
--	--a or a--	Decrements a by 1
4 - Comparison Operators :		
<	a < b	Returns true if a is less than b
>	a > b	Returns true if a is greater than b
<=	a <= b	Returns true if a is less than or equal to b
>=	a >= b	Returns true if a is greater than or equal to b
==	a == b	Returns true if a is equal to b
!=	a != b	Returns true if a is not equal to b
5 - Logical Operators :		
&&	a && b	Returns true if both the expressions are true (non-zero).
||	a || b	Returns true if any of the operands / expressions is true.
!	!a	Returns true/false if operand/expression is false/true.
6 - Bitwise Operators :		
& (Bitwise AND)	A & B	It performs a Boolean AND operation on each bit of its integer arguments. If A holds 2 and B holds 3, then result is 2.
| (BitWise OR)	A | B	It performs a Boolean OR operation on each bit of its integer arguments. If A holds 2 and B holds 3, then result is 3.

^ (Bitwise XOR)	A ^ B	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. If A holds 2 and B holds 3, then result 1.
~ (Bitwise Not)	~B	It is a unary operator and operates by reversing all the bits in the operand. If B holds 3, then result is 4.
<< (Left Shift)	A << 1	It moves all the 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 it by 2, shifting two positions is equivalent to multiplying by 4, and so on. If A holds 2 then result is 4.
>> (Right Shift)	A >> 1	Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. If A holds 2 then result is 1.
>>> (Right shift with Zero)	A >>> 1	This operator is just like the >> operator, except that the bits shifted in on the left are always zero. If A holds 2 then result is 1.
		7 - Miscellaneous Operators :
i) Conditional Operator		
? :	X = (a > b) ? a : b	The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.If Condition (a > b) is true then value of a will be assigned to X otherwise value of b will be assigned to X.
ii) typeof Operator		
typeof	typeof x	The **typeof** operator 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.

JavaScript Statements

JavaScript supports two types of statements – conditional statements and loop statements.

Conditional Statements:

While writing a program, there may be a situation when we need to adopt one out of a given set of paths. In such cases, we need to use conditional statements that allow your program to make correct decisions and perform right actions. JavaScript supports the following conditional statements:

- if statement
- if...else statement
- if...else if... statement
- switch.. case statement

- **if Statement**

The '**if**' statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally. The syntax for a basic if statement is as follows:

```
if (expression){
Statement(s) to be executed if expression is true
}
```

Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed. If the expression is false, then no statement would be executed.

- **if...else Statement**

The '**if...else**' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way. The syntax of an **if-else** statement is as follows:

```
if (expression)
{
Statement(s) to be executed if expression is true
}
else{
Statement(s) to be executed if expression is false
}
```

Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed. If the expression is false, then the given statement(s) in the else block are executed.

- **if...else if... Statement**

The '**if...else if...**' statement is an advanced form of **if…else** that allows JavaScript to make a correct decision out of several conditions. The syntax of an if-else-if statement is as follows:

```
if (expression1)
{
Statement(s) to be executed if expression1 is true
}
else if (expression2){
Statement(s) to be executed if expression2 is true
}
else if (expression3){
Statement(s) to be executed if expression 3 is true
}
else{
Statement(s) to be executed if no expression is true
}
```

It is just a series of **if** statements, where each **if** is a part of the **else** clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the **else** block is executed.

- **switch statement:**

We can use multiple **if...else…if** statements to perform a multi-way branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable. The objective of a **switch** statement is to evaluate an expression and execute different statements based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
switch (expression)
{
case condition1 : statement(s)
          break;
case condition2 : statement(s)
          break;
.
.
case condition : statement(s)
          break;
default :         statement(s)
}
```

The break statements indicate the end of a particular case.

Loop Statements :

While writing a program, we may encounter a situation where we need to perform an action over and over again. In such situations, we would need to write loop statements to reduce the number of lines. JavaScript supports the following loops –

- **The while Loop**

The most basic loop in JavaScript is the **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.The syntax of **while loop** in JavaScript is as follows:

```
while (expression)
{
Statement(s) to be executed if expression is true
}
```

- **The do...while Loop**

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. The syntax for **do-while** loop in JavaScript is as follows:

```
Do {
Statement(s) to be executed;
} while (expression);
```

- **The for Loop**
  The **'for'** loop is the most compact form of looping. It includes the following three important parts:
- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- 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.
- The **iteration statement** where we can increase or decrease our loop counter.

We can put all the three parts in a single line separated by semicolons.
The syntax of **for** loop is JavaScript is as follows:

```
for (initialization; test condition; iteration statement)
{
Statement(s) to be executed if test condition is true
}
```

- **The for ... in loop**

The **for...in** loop is used to loop through an object's properties. The syntax of **'for..in'** loop is:

```
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.

1.10.3 JavaScript Functions

A function is a group of reusable code which can be called anywhere in our program. This eliminates the need of writing the same code again and again. Functions allow a programmer to divide a big program into a number of small and manageable functions. Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions.

Function Definition

Before we use a function, we need to define it. 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.

The basic syntax is shown here.

```
<script language ="JavaScript" type="text/JavaScript">
<!--
function functionname(parameter-list)
{
statements
}
//-->
```

```
</script>
```

Try the following example. It defines a function called **myMsg** that takes no parameters:

```
<script language =”JavaScript” type="text/JavaScript">
<!--
function myMsg()
{
alert("Hello there");
}
//-->
</script>
```

Calling a Function

To invoke a function somewhere later in the script, we simply need to write the name of that function as shown in the following code.

```
<html>
<head> <script language =”JavaScript” type="text/JavaScript">
function myMsg ()
{
document.write ("Hello there!");
}
</script> </head>
<body>
<form>
<input type="button" onclick=" myMsg()" value=" Call myMsg">
</form> </body> </html>
```

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

We have modified our **myMsg** function here. Now it takes two parameters.

```
<html>
<head> <script language =”JavaScript” type="text/JavaScript">
function myMsg (name, age)
{
document.write (name + " is " + age + " years old.");
}
</script> </head>
<body> <form>
<input type="button" onclick=" myMsg('Rita', 7)" value="Call myMsg">
</form> </body> </html>
```

The 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.

The following example defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
<head> <script language =”JavaScript” type="text/JavaScript">
function concatenate(first, last) {
var full = first + last;
return full; }
function secondFunction() {
var result = concatenate('Rita', 'Rani');
document.write (result ); }
</script> </head>
<body> <form>
```

```
<input type="button" onclick="secondFunction()" value="Call Function">
</form> </body> </html>
```

1.10.4 JavaScript Objects

JavaScript is an Object Oriented or object-based Programming (OOP) language. An Object is a set of variables (known as attributes / properties) and functions (known as methods).

- The **String** Object

The **String** object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods. As 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.

The following syntax is used to create a **String** object:

```
var mystr = new String(string);
```

The **string** parameter is a series of characters.

Here is a list of the properties and methods of **String** object and their description.

Property	**Description**
length	Returns the length of the string.
Method	**Description**
charAt()	Returns the character at the specified index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.

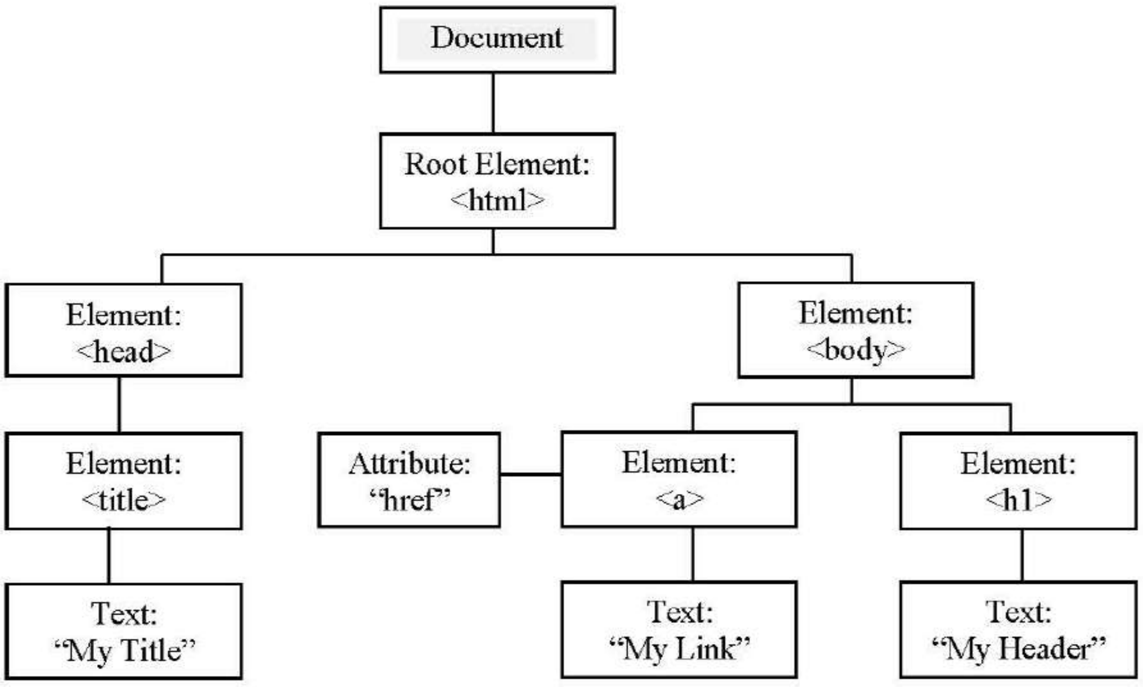
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.

1.11 HTML Document Object Model and JavaScript

As soon as an HTML web page is loaded in the browser, the browser creates a Document Object Model (DOM) of the page. The HTML DOM objects are arranged in a hierarchical (Tree-like) structure.
With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The HTML DOM Tree of Objects



What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard which defines a standard for accessing documents:

"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."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

1.11.1 What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**

- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: "The HTML DOM is a standard for how to get, change, add, or delete HTML elements".

1.11.2 The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages). In the DOM, all HTML elements are defined as **objects**. The programming interface is the properties and methods of each object.

- A **property** is a value that we can get or set (like changing the content of an HTML element).
- A **method** is an action we can do (like add or deleting an HTML element).

1.11.3 The HTML DOM Document Object

The HTML DOM document object is the owner of all other objects in our web page. If we want to access any element in an HTML page, we always start with accessing the document object. Below are some examples of how we can use the document object to access and manipulate HTML.

1.11.3.1 Finding HTML Elements

Often, with JavaScript, we want to manipulate HTML elements. To do so, we have to find the elements first. There are a couple of ways to do this:

Finding HTML elements by id

I. Finding HTML elements by tag name
II. Finding HTML elements by class name
III. Finding HTML elements by CSS selectors
IV. Finding HTML elements by HTML object collections

I. Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id. This example finds the element with id="intro":

```
"color:black">"color:red"> "color:mediumblue">var myElement =
document."color:black">getElementById("color:brown">"intro");
```

If the element is found, the method will return the element as an object (in myElement). If the element is not found, myElement will contain null.

II. Finding HTML Elements by Tag Name

This example finds all <p> elements:

```
"color:black">"color:red"> "color:mediumblue">var x =
document."color:black">getElementsByTagName("color:brown">"p");
```

The next example finds the element with id="main", and then finds all <p> elements inside "main":

```
"color:black">"color:red"> "color:mediumblue">var x =
document."color:black">getElementById("color:brown">"main");
"color:mediumblue">var y =
x."color:black">getElementsByTagName("color:brown">"p");
```

III. Finding HTML Elements by Class Name

If we want to find all HTML elements with the same class name, use get Elements By Class Name(). Following example returns a list of all elements with class="intro".

```
"color:black">"color:red"> "color:mediumblue">var x =
document."color:black">getElementsByClassName("color:brown">"intro");
```

IV. Finding HTML Elements by CSS Selectors

If we want to find all HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method. This example returns a list of all <p> elements with class="intro".

```
"color:black">"color:red"> "color:mediumblue">var x =
document."color:black">querySelectorAll("color:brown">"p.intro");
```

V. Finding HTML Elements by HTML Object Collections

This example finds the form element with id="frm1", in the forms collection, and displays all element values:

```
"color:black">"color:red"> "color:mediumblue">
var x = document."color:black">forms["color:brown">"frm1"];
“color:mediumblue">var text = "color:brown">"";
"color:red"> "color:mediumblue">var i;
"color:mediumblue">
for (i = "color:red">0; i < x."color:black">length; i++) {
  text += x."color:black">elements[i]."color:black">value +
"color:brown">"<br>";
}
document."color:black">getElementById("color:brown">"demo")."color:black">i
nnerHTML = text;
```

The following HTML objects (and object collections) are also accessible:

document.anchors	document.body	document.documentElement
document.embeds	document.forms	document.head
document.images	document.links	document.scripts
document.title		

1.11.3.2 Changing HTML Elements

The HTML DOM allows JavaScript to change the content of HTML elements.

I. Changing the HTML Output Stream

JavaScript can create dynamic HTML content: The document.write() can be used to write directly to the HTML output stream i.e., in document part:

```
<!DOCTYPE<span style="color:red"> html</span="color:mediumblue">>
<html<span style="color:mediumblue"></span>
```

```
<body<span style="color:mediumblue"></span>
<script<span style="color:mediumblue"></span>
document.write(Date());
</script<span style="color:mediumblue"></span>
</body<span style="color:mediumblue"></span>
</html<span style="color:mediumblue"></span>
```

II. Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML = new HTML
```

The following example changes the content of a <p> element:

```
<html<span style="color:mediumblue"></span>
<body<span style="color:mediumblue"></span>
<p<span style="color:red">
id="color:mediumblue">="p1"</span="color:mediumblue">>Hello
World!</p<span style="color:mediumblue"></span>
<script<span style="color:mediumblue"></span>
document.getElementById("p1").innerHTML = "New text!";
</script<span style="color:mediumblue"></span>
</body<span style="color:mediumblue"></span>
</html<span style="color:mediumblue"></span>
```

The next example changes the content of an <h1> element:

```
<!DOCTYPE<span style="color:red"> html</span="color:mediumblue">>
<html<span style="color:mediumblue"></span>
<body<span style="color:mediumblue"></span>
<h1<span style="color:red">
id="color:mediumblue">="header"</span="color:mediumblue">>Old
Header</h1<span style="color:mediumblue"></span>
```

```
<script<span style="color:mediumblue"></span>
var element = document.getElementById("header");
element.innerHTML = "New Header";
</script<span style="color:mediumblue"></span>
</body<span style="color:mediumblue"></span>
</html<span style="color:mediumblue"></span>
```

III. Changing the Value of an Attribute

To change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute = new value
```

Following example changes the value of the src attribute of an <img> element:

```
<!DOCTYPE<span style="color:red"> html</span="color:mediumblue">>
<html<span style="color:mediumblue"></span>
<body<span style="color:mediumblue"></span>
<img<span style="color:red"> id="color:mediumblue">="myImage"</span
src="color:mediumblue">="smiley.gif"="color:mediumblue">>
<script<span style="color:mediumblue"></span>
document.getElementById("myImage").src = "landscape.jpg";
</script<span style="color:mediumblue"></span>
</body<span style="color:mediumblue"></span>
</html<span style="color:mediumblue"></span>
```

Example explained:

- The HTML document above contains an <img> element with id="myImage"
- We use the HTML DOM to get the element with id="myImage"
- A JavaScript changes the src attribute of that element from "smiley.gif" to "landscape.jpg"

1.11.4 DOM Events

The HTML DOM allows us to execute code when an event occurs. A JavaScript code can be executed when an event occurs, like when a user clicks on an HTML element. To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
onclick=JavaScript
```

Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

In following example, the content of the <h1> element is changed when a user clicks on it:

```
<!DOCTYPE<span style="color:red"> html</span="color:mediumblue">>
<html<span style="color:mediumblue"></span>
<body<span style="color:mediumblue"></span>
<h1<span style="color:red"> onclick="color:mediumblue">="this.innerHTML =
'Ooops!'"</span="color:mediumblue">>Click on this text!</h1<span
style="color:mediumblue"></span>
</body<span style="color:mediumblue"></span>
</html<span style="color:mediumblue"></span>
```

HTML Event Attributes

To assign events to HTML elements we can use event attributes.
The following code assigns an **onclick** event to a button element:

```
<button<span style="color:red">
```

```
onclick="color:mediumblue">="displayDate()"</span="color:mediumblue">>Try
it </button <span style="color:mediumblue"></span>
```

In the example above, a function named *displayDate* will be executed when the button is clicked.

Assign Events Using the HTML DOM

The HTML DOM allows us to assign events to HTML elements using JavaScript.

The following code assigns an **onclick** event to a button element using HTML DOM:

```
<script<span style="color : mediumblue"></span>
document.getElementById("myBtn").onclick = displayDate;
</script<span style="color:mediumblue"></span>
```

In the example above, a function named *displayDate* is assigned to an HTML element with the id="myBtn". The function will be executed when the button is clicked.

The **onload** and **onunload** Events

The onload and onunload 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

```
<body<span style="color : red"> onload="color : mediumblue"> =
"checkCookies()"</span="color : mediumblue">>
```

The **onchange** Event

The **onchange** event is often used in combination with validation of input fields. Below 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

```
<input<span style="color : red"> type="color : mediumblue">="text"</span
id="color : mediumblue">="fname "onchange="color :
mediumblue">="upperCase()" = "color:mediumblue">>
```

The **onmouseover, onmouseout, onmousedown, onmouseup and onclick** Events

The **onmouseover** and **onmouseout** events can be used to trigger a function when the user mouses over, or out of, an HTML element. The **onmousedown**, **onmouseup**, and **onclick** 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.

1.12 Summary

- HTML is a tag-based language generally used to create statics web pages, therefore web pages are also referred to as HTML documents.
- A Dynamic web page is one where the structure, style, or content of the page can be changed after the page is loaded in the browser.
- Style sheets are documents where we can define styles and use them in different pages of web site.
- A Form is a collection of fields that can be used to gather information from the visitors of the web site. Forms act as a means of user interaction on the web.
- JavaScript is a dynamic, script-based, light-weight, interpreted computer programming language with object-oriented capabilities. JavaScript supports the development of both Client and Server components of web-based applications.
- 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."

1.13 Self Assessment Questions

Q.1 Discuss tags in HTML Head section.

Q..2 List tags and attributes for HTML Table.

Q..3 List Frame and Frameset tags and their attributes.

Q..4 Discuss DOM and its Hierarchy.

Q.5 Discuss Text structuring tags and attributes.

Q.6 Discuss methods for linking pages available in Javascript.

Q.7 Discuss String object of Javascript.

References and Suggested Reading

1. N.P.Gopalan, J. Akilandeswari, "Web Technology - A Developer's Perspective", PHI Learning Private Limited.

2. Kogent Learning Solutions Inc., "Web Technologies: HTML, JAVASCRIPT, PHP, JAVA, JSP, ASP.NET, XML and Ajax, Black Book: HTML, Javascript, PHP, Java, Jsp, XML and Ajax, Black **Book",** Dreamtech Press.

3. Ivan Bayross, "Web Enabled Commercial Application Development Using HTML, DHTML, PERL, Java Script", BPB Publications.

UNIT-2
Events Handlers & Forms in JavaScript

Structure of the Unit

2.0 Objective

In this chapter, we shall focus on the following topics

- Events and associated event handlers
- Mouse and keyboard events
- Emulating events
- Web hoping
- Dealing with popup windows

2.1 Introduction

In the current scenario, it is required to know how to write a program in event driven mode. Event driven programming is used to develop interactive, attractive and user-friendly web applications. Website application development is twofold - client side and server side. HTML is used for client side development in which user interfaces are designed and developed. In a nutshell, the appearance of the website, web pages and web forms, are designed in client side programming using HTML. On the other hand, server-side programming provides functionality to these web pages. How will a web page react if an event has occurred? The answer to this question is provided by JavaScript or other scripting or programming languages in server side code. For each event triggered on the client side, there is an associated event handler in server side scripting language. This event handler will decide that how the web page will react in response to an event. A detailed discussion is given on the events, event handlers and event handling in upcoming headings.

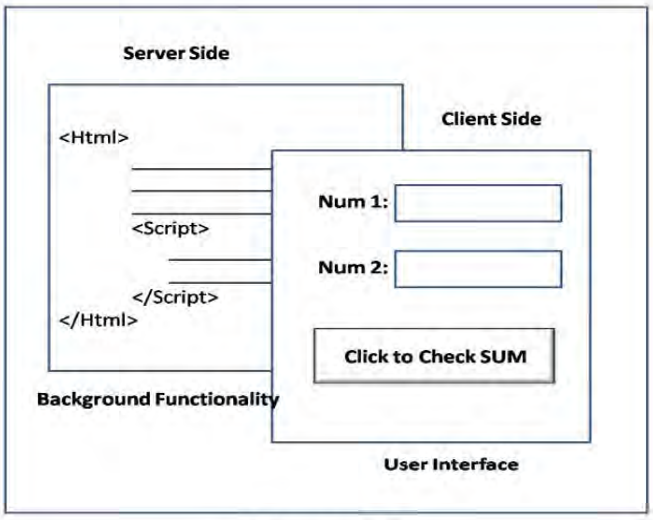
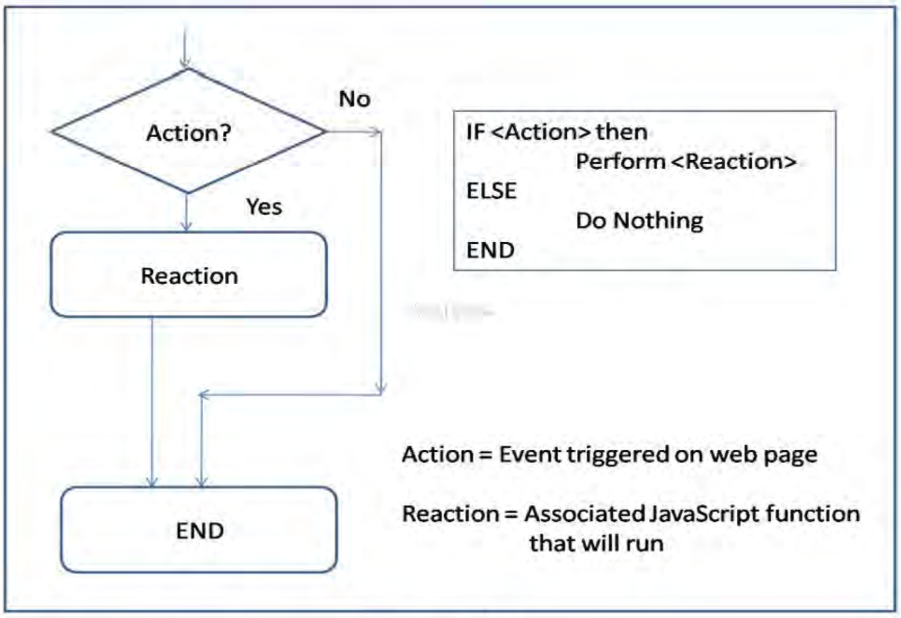


Figure 2.0

2.2 Define Events

HTML is used to design the structure of a web page. HTML code will decide that how the web page will look like when opened in a web browser. JavaScript provide functionality to HTML-based web pages. HTML is responsible for static web pages whereas JavaScript provides facility to create dynamic web pages. A web page is considered as dynamic if it can respond/react if an activity has been performed on it. These activities, performed on web pages are called events. Loading a web page in browser, minimising or maximising a web page, clicking a button or loading an image etc. is an example of an event. An event is the occurrence of some incident at a particular time. What will happen when a button is clicked or when a web page will be minimised; this will be decided by the JavaScript code. This way, JavaScript and HTML work together. JavaScript interacts with HTML page through events and events can be triggered when the user or the browser manipulates a page.


Action?
No
Yes
Reaction
END
Action = Event triggered on web page
Reaction = Associated JavaScript function that will run

```
IF <Action> then
        Perform <Reaction>
ELSE
        Do Nothing
END
```

Figure 2.1

Figure 2.1 is showing the process of event triggering. The essence of this figure is that there is always a reaction for an action (event). Reaction process is handled by a special function in JavaScript that is called event handler.

2.3 Events in JavaScript

Events are a part of the Document Object Model (DOM) and every HTML element contains a set of events that can trigger a JavaScript Code. Some events get fired automatically like loading a page, but some events need to be triggered by the user e.g. clicking on a button.

Events are of two types-bowser driven and user driven. These two types of events are shown in figure 2.2. The browser driven events are automatically triggered by the programmer. On the other hand, the user-driven events depend on an activity of the user on the web page.

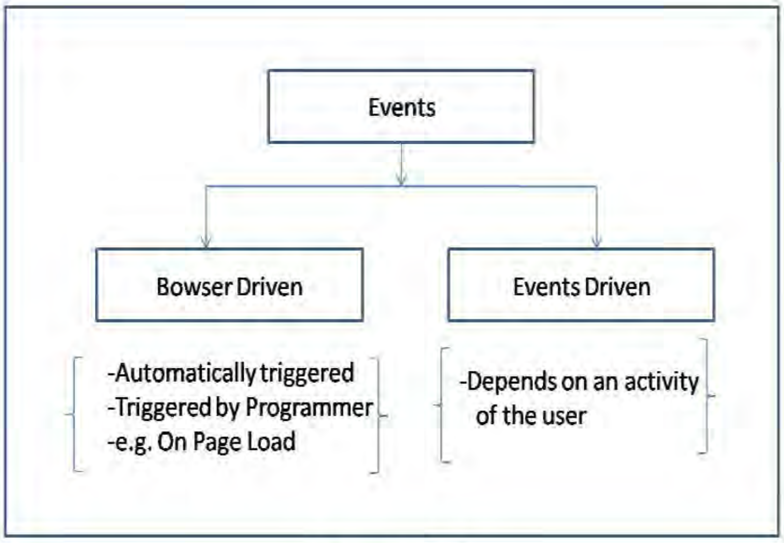


Figure 2.2

When a page is loaded in a browser, so many events get triggered automatically or on a reaction of the user. Some of these triggered events are those we are interested in like clicking on a button, stopping a page etc. we really not have interest in some of the events like when a user move mouse on the webpage or an element; it will trigger onmouseover and onmousemove events that may not be of our interest. It is the responsibility of a programmer to listen and react to only those events in which he is actually interested in.

2.4 Event Handlers

JavaScript interacts with HTML through events that occur when the user or the browser manipulates/interacts with a page. Events are normally used in combination with functions. Scripts to be executed when they are called, or when an event is triggered, are placed in functions. It is a convention to put these functions in the head section. This way they are all in one place, and they do not interfere with page content.

JavaScript code has been executed when an event is triggered. To work with events there are two steps to do-

a) Listen for events

b) Respond to events

a) Listening for Events

Listening to right / desired events is handled by an especial function called addEventListener(). This function is responsible for keeping an eye on the web page and notify when the desired event gets fired. The exact syntax of this function is –

```
source.addEventListener(eventName, eventHandler, false/ true)
```

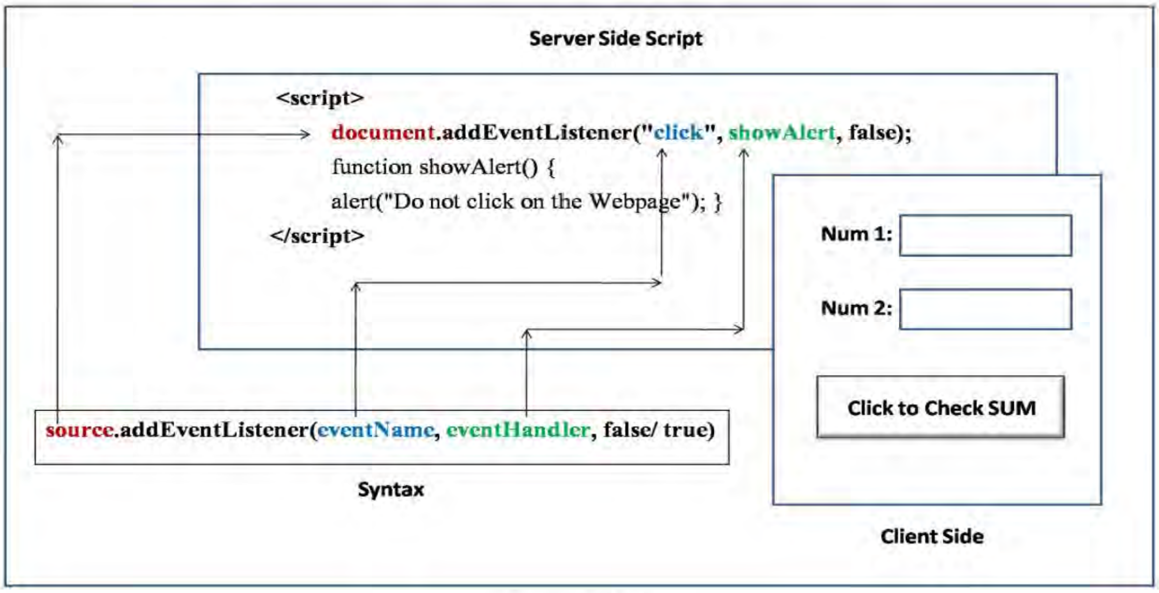

Server Side Script
<script>
document.addEventListener("click", showAlert, false);
function showAlert() {
alert("Do not click on the Webpage"); }
</script>
source.addEventListener(eventName, eventHandler, false/ true)
Syntax
Num 1:
Num 2:
Click to Check SUM
Client Side

Figure 2.3

In the above syntax, the source is any object that fires the event. It can be DOM element, window, document or object on a web form. In argument list, eventName is the name of the event we are interested in listening to. The eventHandler is the name of the function that will be called on the trigger of the event mentioned in eventName. The last argument is to mention whether we want to listen to the event during capturing phase or bubbling phase. The capturing and bubbling phases of events are explained in the next section. The default value of this argument is false which indicate bubbling phase. Figure 2.3 is showing the relation between syntax and the actual code of event listener.

Code for listening events is written in example 2.1 at line number 7.

```
document.addEventListener("click", showAlert, false);
```

Here source documents itself. Click event is to listen in bubbling phase and function to be called in response to this event is with showAlert() method.

b) Responding to events

JavaScript responds to the events by calling respective functions mentioned in the event handler. In Example 2.1, a function with name showAlert() is associated with the click event given in the event handler written at line 8 in example 2.1. So showAlert() function is automatically called on clicking the document.

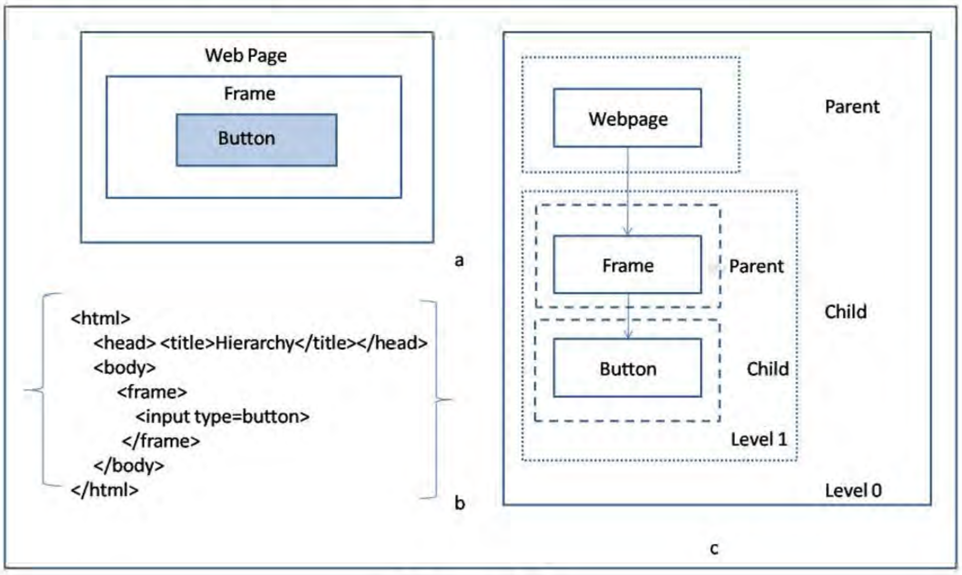
Example 2.1

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <title>Click Anywhere!</title>
5   </head>
6   <body>
7    <script>
8   document.addEventListener("click", showAlert, false);
9   function showAlert() {
10  alert("Do not click on the Webpage"); }
```

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

Capturing and bubbling phase

DOM elements in the JavaScript are nested in each other. Outer element is called parent and elements inside parent are called its child. The events handlers of these elements are also triggered in nested form. If we trigger click event on a child element (Button in figure 2.4(a)), the handler of the parent will also work. To understand this, it is necessary to know the hierarchy of a document and life cycle of an event. Consider following figure to understand this. The figure a and b is showing the hierarchy of elements in a document. The hierarchy is – webpage->frame->button. The frame is a child of the web page (level 0) and the frame itself is a parent of a button(level 1) (figure 2.4 c).



```
<html>
  <head> <title>Hierarchy</title></head>
  <body>
    <frame>
      <input type=button>
    </frame>
  </body>
</html>
```

Figure 2.4

When a click event is triggered on button element, capturing and bubbling phases will be triggered. Capturing and bubbling are two phases of an event. All browsers other than Internet Explorer following this two-phase life cycle of an event.

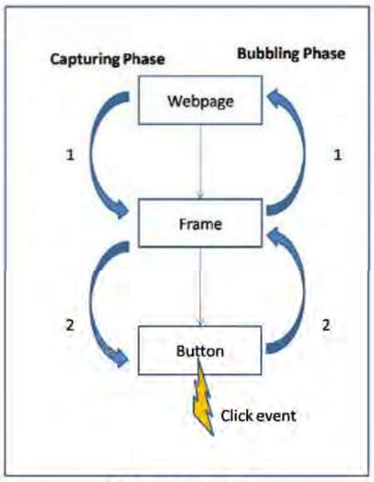


Figure 2.5

Consider the example in figure 2.5. Even though the click event is triggered on the button, the life cycle of this click event will start from the webpage. First capturing phase is generated and then after bubbling phase is generated.

Capturing phase- Webpage -> Frame -> Button (1->2)

Bubbling phase – Button -> Frame -> Web Page (2->1)

The process is shown in figure 2.5.

All the methods of event handling ignore the capturing phase and wait for bubbling phase to reach (by default). To catch an event at capturing phase, we have to pass true in the last argument in addEventListener. If phase = true is passed then the handler is set to the capturing phase and if phase = false is passed then The handler is set in the bubbling phase.

```
element . addEventListener ( type, handler, phase )
```

For Internet Explorer <9, event.cancelBubbling = true property is used instead of phase argument.

Event object and event type

An object of the event is created in JavaS on triggering. This object can provide some additional information about the event triggered. This object can be passed to the associated function that will be called as the reaction of the event. All events originate from a common base Event (an Interface) so they have some common properties. Events also have some unique properties. Some properties of events are-

a) currentTarget
b) target
c) preventDefault
d) stopPropagation
e) type etc.

Removing Event Listener

Event Listener can be neutralised at any time of execution by writing following line in the code snippet-

```
        Source.removeEventListener("eventName, eventHandler, false/ true");
```

If we want to neutralise the effect of event listener in Example 2.1 then we can add following line in the code after event listener.

```
        <script>
            document.addEventListener("click", showAlert, false);
            function showAlert() {
            document. removeEventListener ("click", showAlert, false);
            alert("Do not click on the Webpage"); }
        </script>
```

Different ways to write Event Handlers

There are two different ways to write event handlers. Suppose there is a text box on a form and we want to validate the inputted values. This will trigger the onchange event of text field. So we have to write code-

```
<INPUT TYPE="text" onChange="validateInput(this)">
function validateInput(this)
{
  If(parseInt(this.value)>10){
      alert('Please enter a number less than 10');}
```

Another method to perform same is to embed all code in double quotes-

```
<INPUT TYPE="text" onChange="
  If(parseInt(this.value)>10){
       alert('Please enter a number less than 10');}">
```

The advantage of using functions as event handlers is that the same event handler code can be used for multiple items in one document and functions make it easier to read and understand and debug the code.

2.5 This keyword

The this keyword refers to the current object. In the case of

```
<INPUT TYPE="text" onChange="checkField(this)">
```

this refers to the current text object. In JavaScript, forms and forms elements are objects. These form elements include text fields, checkboxes, radio buttons, buttons, and selection lists. The this keyword refers to the element which has triggered the code.

2.6 Event handlers in JavaScript

HTML DOM events allow JavaScript to register different event handlers on the elements in HTML document. Providing a list of JavaScript event handlers-

Mouse Event Handlers

Event	Description	DOM
onclick	The event occurs when the user clicks on an element	2
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu	3
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when the user presses a mouse button over an element	2

onmouseenter	The event occurs when the pointer is moved onto an element	2
onmouseleave	The event occurs when the pointer is moved out of an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
onmouseup	The event occurs when a user releases a mouse button over an element	2
onclick	The event occurs when the user clicks on an element	2

Keyboard Event Handlers

Event	Description	DOM
onkeydown	The event occurs when the user is pressing a key	2
onkeypress	The event occurs when the user presses a key	2
onkeyup	The event occurs when the user releases a key	2

Frame/Object Event Handlers

Event	Description	DOM
onabort	The event occurs when the loading of a resource has been aborted	2
onbeforeunload	The event occurs before the document is about to be unloaded	2
onerror	The event occurs when an error occurs while loading an external file	2
onhashchange	The event occurs when there has been changes to the anchor part of a URL	3
onload	The event occurs when an object has loaded	2

onpageshow	The event occurs when the user navigates to a webpage	3
onpagehide	The event occurs when the user navigates away from a webpage	3
onresize	The event occurs when the document view is resized	2
onscroll	The event occurs when an element's scrollbar is being scrolled	2
onunload	The event occurs once a page has unloaded (for <body>)	2

Form Event Handlers

Event	Description	DOM
onblur	The event occurs when an element loses focus	2
onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <keygen>, <select>, and <textarea>)	2
onfocus	The event occurs when an element gets focus	2
onfocusin	The event occurs when an element is about to get focus	2
onfocusout	The event occurs when an element is about to lose focus	2
oninput	The event occurs when an element gets user input	3
oninvalid	The event occurs when an element is invalid	3
onreset	The event occurs when a form is reset	2
onsearch	The event occurs when the user writes something in a search field (for <input="search">)	3
onselect	The event occurs after the user selects some text (for <input> and <textarea>)	2
onsubmit	The event occurs when a form is submitted	2

Drag Event Handlers

Event	Description	DOM
ondrag	The event occurs when an element is being dragged	3
ondragend	The event occurs when the user has finished dragging an	3

	element	
ondragenter	The event occurs when the dragged element enters the drop target	3
ondragleave	The event occurs when the dragged element leaves the drop target	3
ondragover	The event occurs when the dragged element is over the drop target	3
ondragstart	The event occurs when the user starts to drag an element	3
ondrop	The event occurs when the dragged element is dropped on the drop target	3

Event Object

Constants

Constant	**Description**	**DOM**
CAPTURING_PHASE	The current event phase is the capture phase (1)	1
AT_TARGET	The current event is in the target phase, i.e. it is being evaluated at the event target (2)	2
BUBBLING_PHASE	The current event phase is the bubbling phase (3)	3

Properties

Property	**Description**	**DOM**
bubbles	Returns whether or not a specific event is a bubbling event	2
cancellable	Returns whether or not an event can have its default action prevented	2
currentTarget	Returns the element whose event listeners triggered the event	2
defaultPrevented	Returns whether or not the preventDefault() method was called for the event	3
eventPhase	Returns which phase of the event flow is currently	2

	being evaluated	
isTrusted	Returns whether or not an event is trusted	3
target	Returns the element that triggered the event	2
timestamp	Returns the time (in milliseconds relative to the epoch) at which the event was created	2
type	Returns the name of the event	2
view	Returns a reference to the Window object where the event occured	2

Methods

Method	Description	DOM
preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur	2
stopImmediatePropagation()	Prevents other listeners of the same event from being called	3
stopPropagation()	Prevents further propagation of an event during event flow	2

A detailed discussion of mouse and keyboard events and their properties is given underneath-

The MouseEvent Properties

All mouse events are based on the MouseEvent object. The MouseEvent has associated with it a list of very important properties.

1. **screenX and screenY** – These properties returns the distance of mouse cursor from the top-left corner of the monitor.
2. **clientX and clientY**- These properties returns the x and y position of the mouse relative to your browser's top-left corner

   The difference between these two properties is shown in figure 2.6.

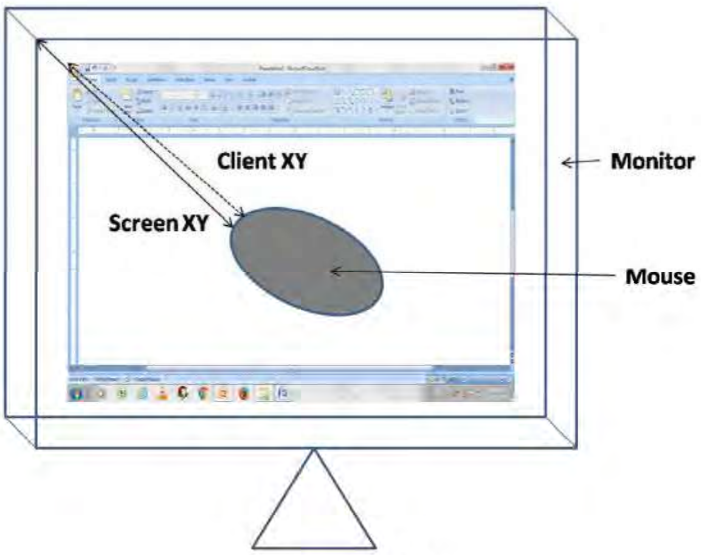


Figure 2.6

3. **button**- This property is used to figure out the button pressed by the user. There exist three buttons in a common mouse- left button, right button and a middle button. Following values are returned by button property-

   Left mouse button click event – 0

   Middle mouse button click event – 1

   Right mouse button click event – 2

Example 2.2- mouse button event

```
<!DOCTYPE html>
<html>
   <head>
          <title>Example for Mouse Button Click!</title>
   </head>
   <body>
         <script>
               document.addEventListener("mousedown", buttonPress, false);
               function buttonPress(e) {
               if (e.button == 0) {
                  alert("Left mouse button pressed!");
```

```
                    } else if (e.button == 1) {
                        alert("Middle mouse button pressed!");
                    } else if (e.button == 2) {
                        alert("Right mouse button pressed!");
                        }}
        </script>
    </body>
</html>
```

Dealing with the Mouse Wheel

There is a special event called mouse wheel event which shows mousewheel property. Different names are used for mousewheel event in different browsers. Internet Explorer and Chrome have mousewheel event to deal with the mouse wheel. On the other hand, Firefox uses DOMMouseScroll event for dealing mouse wheel. The mousewheel and DOMMouseScroll events will fire when user scroll the mouse wheel in any direction. An argument is passed in the mousewheel event known as wheelDelta property. The DOMMouseScroll event has detail property in the event argument. Both of these properties are similar in that their values change from positive or negative depending on what direction of scroll the mouse wheel. The wheelDelta property associated with the mousewheel event is positive for scroll up on the mouse wheel and it is negative for scroll down. The DOMMouseScroll's detail property holds exact opposite values negative for scroll up and positive for scroll down. The scrollDirection variable stores the value contained by the wheelData property or the detail property.

The Keyboard Event Properties

An argument for keyboard event is passed in keyboard event handler is called keyboard event argument. For example, see the following event handler for dealWithKeyboard event handler. The keyboard event is represented by the e argument that is passed in:

```
function dealWithKeyboard(e) {
```

```
// gets called when any of the keyboard events are overheard }
```

This argument e contains following important properties:

Following are the properties of *Keyboard* event.

keyCode -

Each key on a keyboard has a unique code to detect it. This property returns key code of the key that is pressed. This is a read-only property.

charCode

This property contains the ASCII code for the key pressed. The charCode and keyCode values for a particular key are not the same. The charCode is only returned if the event that triggered the event handler was keypress.

ctrlKey, altKey, shiftKey

These three properties return a true if the Ctrl key, Alt key, or Shift key are pressed.

metaKey

This property returns a true if the Meta key is pressed. The Meta key is the Windows key on Windows keyboards and the Command key on Apple keyboards.

Example 2.3: To check if a particular key was pressed

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
        window.addEventListener("keypress", checkKeyPressed, false);
        function checkKeyPressed(e) {
        if (e.charCode == "97") {
            alert("The 'a' key is pressed."); }}
</script>
</html>
```

Example 2.4 : Arrow Key Press

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
        document.onkeydown = function(e) {    switch (e.keyCode) {
        function checkKeyPressed(e) {
        if (e.keyCode == "37") {
            alert("The Left arraow key is pressed."); }}
</script>
</html>
```

2.7 Emulating Events in java scripting

It is possible to emulate events through JavaScript. This emulating process is useful to submit a form without user intervention. Emulation of events is generally used to validate data on a form or to perform confirmation of submission of a form.

Suppose you are designing an online submission form of an institution. It is required to fill all details by the student and otherwise form should not be submitted. In this case once student fills the required details in the online form and click on the submit button; the form should not be directly sent to the institutional server. Once user clicks on the Button, instead of sending info directly to the server, it will check whether the form is filled as per requirement or not. If form is ok then JavaScript will trigger click event implicitly. Otherwise JavaScript set the focus on form again. The process of implicit calling of events is called emulating events. There are some event methods available in JavaScript which are useful for this purpose- blur(), click(), focus(), reset(), select() and submit(). Events generated with these methods do invoke their corresponding event handlers.

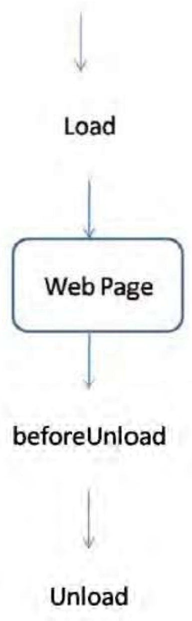


Figure 2.7

2.8 onLoad and onUnload Event Handlers

There are three important events of a web page. These are – load, unload and beforeunload. The relationship between these events is shown in figure 2.7. The load event of a page has been generated when the page has completed the process of loading in the memory. An onload event handler is a good place for initialization. There is a minor problem with onload event handler. Because onload executes when everything is loaded, a visitor has to wait for all images etc to complete loading. In modern versions of browsers, a special event method DOMContentLoaded has been introduced. It triggers when the HTML page is parsed and DOM tree is built. It doesn't wait for images and style sheets. The unload event has been generated When the user exits a page. The unload event happens when a web page is closed or changes its location. The onunload event handler enables a script to be executed before new page loads. Just before the unload event, beforeunload event has been generated. The onbeforeunload event is special because it triggers before the window is closed or changes location. Before unloading the window, the browser will output the returned text and ask the user, if he really wants to leave the page. The onbeforeunload event is useful for this purpose. The web page can check if the document is saved and ask the user if he really wants to leave without saving the job.

Example 2.5: onload and onunload event handlers

```
<!DOCTYPE html>
<html>
<body onLoad="alert('Page is Loading!');"
        onBeforeUnload="alert('Before UnLoading!');"
         onUnload="alert('byebye!!');">
    <img src="title.jpeg">
</body>
</html>
```

2.9 Web-Hopping with window.open()

Window.open() method is used to generate popup windows. Popup windows are used to show the active content of a page or to open a new webpage in a new window.

The window.open() method takes three arguments. The syntax of window.open() is given below-

Syntax :

```
Window.open(url, name, params)
```

where:

The first argument, url is an URL to be loaded into the new window. The second argument, the name is a name of the new window. These two arguments are mandatory. The third argument is params which are a configuration string for the new window. It contains settings, delimited by a comma. There must be no spaces in params. There are two types of param settings; for window position and for window features. Position of window will be mentioned by left/top (numeric) and width/height (numeric) parameters. The left/top (numeric) parameters are coordinates of the window top-left corner on the screen. The width/height (numeric) parameter give width and height of a new window. Parameters for Window features: menubar (yes/no) shows or hides the browser menu on the new window, toolbar (yes/no) shows or hides the browser navigation bar (back, forward, reload etc) on the new window, location (yes/no) shows or hides the URL field in the new window, status (yes/no) shows or hides the status bar, resizable (yes/no) allows to disable the resize for the new window, scrollbars (yes/no) allows to disable the scrollbars for the new window.

Example 2.6 calls window.open without params and example 2.7 calls windows.open with params.

Example 2.6 – window.open() without params

```
<!DOCTYPE html>
<html>
<script language="JavaScript">
        window.open("https://www.yahoo.com/","YahOOOO");
</script>
</html>
```

Example 2: windows.open() with params

```
<!DOCTYPE html>
<html>
<script language="JavaScript">
    var p1 = 'scrollbars=no,resizable=no,status=no,location=no,toolbar=yes,menubar=yes';
    var p2 = 'width=0,height=0,left=-1000,top=-1000';
    window.open("https://www.yahoo.com/","YahOOOO",p1+p2);
    window.open("/","test", p1+p2);
</script>
</html>
```

2.10 Resetting Event Handlers

Event handlers can be explicitly reset in Navigator 3.0. The setting expression of an event handler is evaluated when an event has occurred. This is possible to reset the event handler to a new expression. The new expression to be evaluated takes the form of a function call. For example, if the onClick event handler in the <INPUT> tag of the form named myForm is defined as

```
<INPUT TYPE=button NAME="myButton" onClick="firstFunction()">
```

It is not neccesory to be true for all the time. We can reset this expression for its subsequent calls.

```
document.myForm.myButton.onclick = secondFunction;
```

Now, the subsequent clicks on the button would launch secondFunction() instead of firstFunction().

2.11 Self Learning Exercise

Q.1 In general, event handler is nothing but-

- a. function
- b. interface
- c. event
- d. handler

Q. 2 When will the browser invoke the handler?

- a. The program begins
- b. Any event occurs
- c. Specified event occurs
- d. None of the mentioned

Q.3 Which event is fired when a document and all of its external resources are fully loaded and displayed to the user?

a. Window

b. Load

c. Element

d. Handler

2.12 Summary

Events handlers are functions that will invoke when the associated event is triggered. There are several HTML/DOM events like document, window, mouse keyboard events etc. There is relevant and associated event handler for each event. The event can be caught in capturing or bubbling phase. It is also possible to emulate events. The window.open() is used for web hopping purpose by which popup window effect can be generated.

2.13 Glossary

Event—an incident of something happening

Event Handler- a special function that will be called on the occurrence of an event

Web hopping- opening of a new web page in the same or a popup window

Emulating events- trigger event through code

DOM- Document Object Model

2.14 Answers to Self-Learning Exercise

Q.1	(a)
Q.2	(b)
Q.3	(b)

2.15 Exercise

Q. 1 Add an input box and a button to the page. Write Java Script code snippet for onclick() event handler which gets the value from an input box and computes and convert it to paisa.

Q. 2 Design a page to calculate a bill of a pizza order. A small pizza is Rs 120/-, a medium pizza is for Rs 220/- and a large pizza is for Rs. 320/-. Tax is an additional 9% to the order. The user must also pay a tip from 10-20% of the total cost of the order after tax. Write the JavaScript code necessary to calculate and display the total cost of the order.

Answers of Exercise

Q1.

```
<!DOCTYPE html>
<html>
<script>
        function calculate()
        {
                var rs=parseInt(amount.value);
                alert(rs*100);
        }
</script>
<body>
<h1>Calculate your change</h1>
<fieldset>
Amount:
<input type="text" id ="amount" />
<button onclick="calculate();">Calculate</button>
</fieldset>
```

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

Q2:

```
<!DOCTYPE html>
<html>
<script>
          function calculate()
          {
                   var s=parseInt(small.value);
                  var m=parseInt(medium.value);
                    var l=parseInt(large.value);
                    var tip;
                    if(ten) {tip=10;}
                    if(fifteen) {tip=15;}
                    if(twenty) {tip=20;}
                    var bill=(s*120+m*220+l*320);
                    bill=((bill*9)/100);
                    bill=((bill*tip)/100);
                    alert(bill);
              }
</script>
<body>
    <h1>Pizza Order Form</h1>
    <fieldset>
        # of Small Pizzas <input type="text" id="small" /><br />
       # of Medium Pizzas <input type="text" id="medium" /><br />
       # of Large Pizzas <input type="text" id="large" /><br />
Tip:
<label><input type="radio" id="ten" name="tip" /> 10%</label>
<label><input type="radio" id="fifteen" name="tip" checked="checked" />
```

```
15%</label>
<label><input type="radio" id="twenty" name="tip" /> 20%</label> <br />
button id="submit" onclick="calculate();">Calculate Order</button>
</fieldset>
</body>
</html>
```

References and Suggested Readings

1. JavaScript, A Beginner's Guide, Third Edition by John Pollock, O'Reilly Media, March 2014.
2. Head First JavaScript Programming A Brain-Friendly Guide by Eric T. Freeman, Elisabeth Robson, Tata McGraw Hill, 2008.
3. JavaScript: A Beginner's Guide by John Pollock, McGraw-Hill/Osborne, 2004.
4. JavaScript Tutorial - W3Schools www.w3schools.com/js/

UNIT-3
Messaging & Timing Events in JavaScript

Structure of the Unit

3.0 Objective

In this chapter we shall focus upon the following topics-

- Dialog boxes
- Timing events

3.1 Introduction

Messaging is an important tool for communication of JavaScript with the user. There are three special types of dialog boxes which can commonly be used in JavaScript as a communication tool. These are an alert box, confirm box and prompt box. The use and purpose of these three boxes are different and specific. The detailed description of these messaging boxes is given in following titles.

3.2 Alert Box: syntax & its example

Alert box is used when information is required to be conveyed to the users. Alert boxes are used as a one-way messaging systems where JavaScript can show his message to user but user can not interact with it. When an alert box pops up, the user will have to click "OK" to proceed.

Suppose a web page has been intermitted by the user or some third party tool and it is aborting forcefully in this case alert box can be used to float information. Or if a webpage cannot be loaded because of the interruption in internet connection, the alert box can be used.

The alert dialog box in JavaScript is a method of window object that is available in JavaScript1.0+ versions. It is also available Jscript1.0, Netscape Navigator 2+, Internet Explorer 3+ and Opera3+ versions. Syntax for this is given as under-

Syntax-

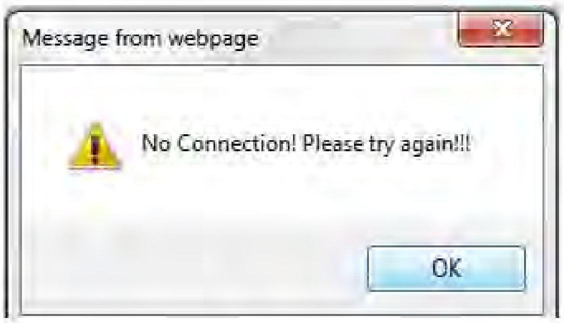
```
Window.alert(String)
```

The String passed in the alert() method is a message that is displayed in the dialog box. This method will not return anything to the JavaScript.

Example 3.1: alert box

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
        alert("No Connection! Please try again!!!");
        </script>
    </html>
```

Output Example 3.1:



No Connection! Please try again!!!

3.3 Confirm Box: syntax & its example

The confirm dialog box is a two-way communication. JavaScript shows a message to the user in this dialog box. The user can respond to this message by clicking on ok or cancel buttons on it. This is also called confirmation box.

The confirm box can be used when the web page is going to perform some action but need an approval from the user. Suppose the close button on the web page is clicked by the user. In this case, a confirm box is a reasonable tool to ask the user whether he really want to close the web page or it is an accident. Or when a customer deleting an item from its cart, confirmation is required and could be used.

Confirm dialog box in JavaScript is a method of window object that is available in JavaScript1.0+ versions. It is also available in Jscript1.0, Netscape Navigator 2+, Internet Explorer 3+ and Opera3+ versions. Syntax for this is given as under-

Syntax-

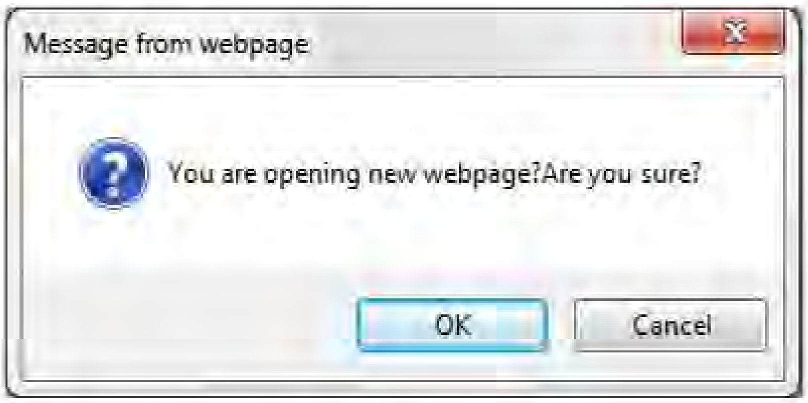
```
window.confirm(String)
```

The String passed in the alert() method is a message that is displayed in the dialog box. The box will contain both OK and Cancel button. This method will return a Boolean value. It returns true when the user clicks on OK and return false when the Cancel button is clicked.

Example -3.2: confirm box

```
<!DOCTYPE html>
<html>
    <script type="text/javascript">
        var ans=confirm("You are opening new webpage? Are you sure?");
        if(ans==true){alert("Yes... You want to continue");}
        else{alert("So sad... Why?");}
    </script>
</html>
```

Output Example 3.2:



You are opening new webpage?Are you sure?

3.4 Prompt Box: syntax & its example

The prompt box is a dialog box with a text box to enter some value. This is used to get some input from the user. prompt() is a method of the Window object. The value entered in the text field of the prompt box is returned to the window. 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. The syntax of prompt box is given below-

Syntax-

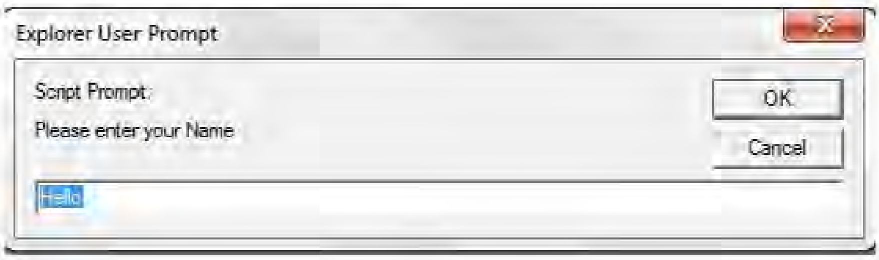
window.prompt(Message, Default)

There are two arguments to be passed in the prompt method. The first is the message that will be displayed in the prompt box. The second argument is the default value which is optional. This default value will be displayed in the text box of the prompt box if passed.

Example 3.3: Prompt box

```
<!DOCTYPE html>
<html>
        <script type="text/javascript">
                var name=prompt("Please enter your Name","Hello");
                alert(name);
        </script>
</html>
```

Output Example 3.3:


Explorer User Prompt

Script Prompt:

Please enter your Name

Hello

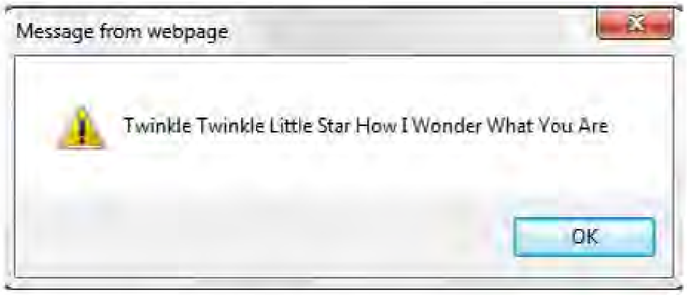
3.5 Line Breaks: syntax & its example

A back-slash followed by the character n (\n) is used to display line breaks inside a popup box for example in an alert box. Normally alert box show string in a single line. If the string is long, then it is awkward to show complete string in an alert box in a single line. In that case, line breaks are used. it is to enhance readability and presentation of the information.

Example 3.4 : message without line break

```
<html>
<head>
        <script type="text/javascript">
                alert("Twinkle Twinkle Little Star How I Wonder What You Are");
        </script></head>
      <body>Welcome</body>
</html>
```

Output Example 3.4:

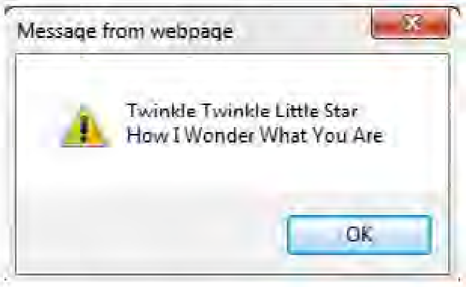


Twinkle Twinkle Little Star How I Wonder What You Are

Example 3.5 : message without line break

```
<html>
<head>
        <script type="text/javascript">
              alert("Twinkle Twinkle Little Star \n How I Wonder What You Are");
        </script></head>
      <body>Welcome</body>
</html>
```

Output Example 3.5:


Message from webpage

Twinkle Twinkle Little Star
How I Wonder What You Are

It is important to know that the newline character can only break in the source code, or in a dialog box. The newline character cannot create a break in the rendered HTML. To create a line break in rendered HTML, the HTML break tag can be used.

3.6 JavaScript Timing Events

JavaScript code can be executed at specified time intervals or gaps these time intervals are called time events. These time events can be applied on HTML DOM window object. There are four special time event methods of window object -

1. setTimeout()
2. setInterval()
3. clearTimeout()
4. clearInterval()

These methods are given in detail in later headings.
Before proceeding, it is important to understand following two characteristics of JavaScript–

1. **JavaScript is single-threaded**
2. This means that the JavaScript window can perform only one operation/ handle one event/ execute one function at a time.
3. **JavaScript events are asynchronous**
4. This means that the events will execute in sequence. Most of the JavaScript events are asynchronous. Most actions occur asynchronously and create an event which is appended to the Event queue. They are taken from the queue

and processed when the time permits. Many events are integrated with JavaScript and many events are strictly internal. The browser has an inner loop, called Event Loop, which checks the queue and processes events, executes functions etc.For example, if the browser is busy processing your onclick, and another event happened in the background (like script onload), it appends to the queue. When the onclick handler is complete, the queue is checked and the script is executed. setTimeout/setInterval also put executions of their functions into the event queue if the browser is busy. In fact, most interactions and activities get passed through the Event Loop.

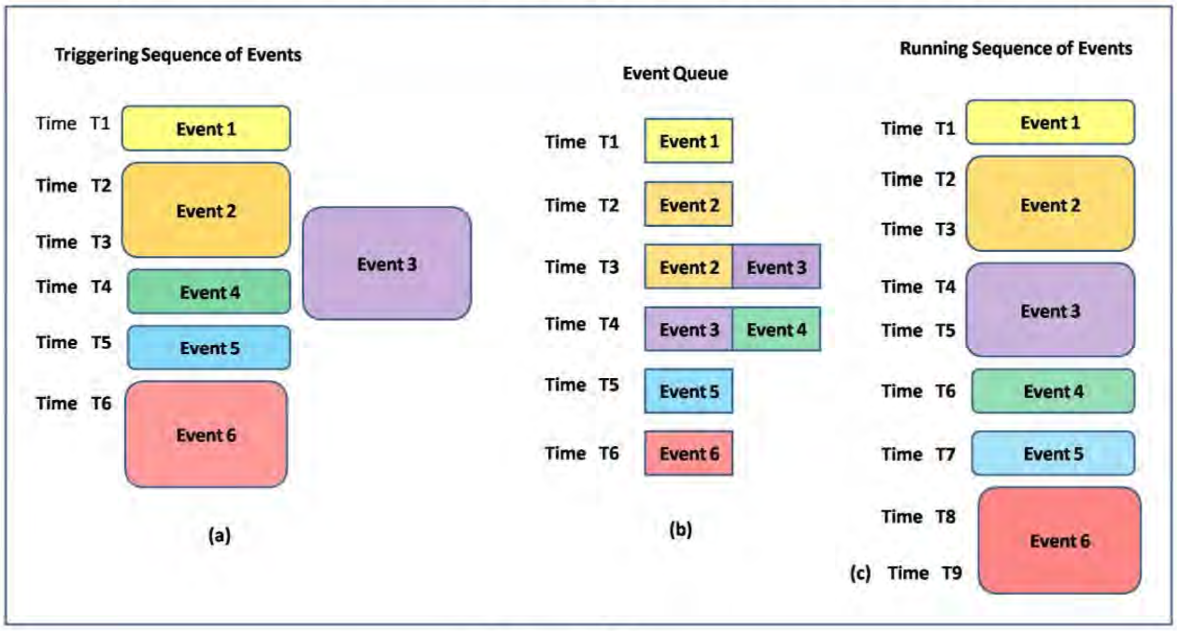


Figure 3.1

Figure 3.1 is showing the triggering of a sequence of events. In figure 3.1 (a), Event 1 is triggered at T1 and the total execution time of Event1 is one unit of time only. Event 2 is triggered at T2 and total execution time required for Event 2 is two unit of time T2 and T3.

3.7 setTimeout() Method

SetTimeout() method executes a function, after waiting a specified number of milliseconds. It initiates a single timer which will call the specified function after the delay. The function returns a unique ID with which the timer can be cancelled at a later time. This ID can be used for clearTimeout() method. The setTimeout() method is a one-time process that is not repeated an infinite number of times.

Syntax:

```
window.setTimeout(function, milliseconds)
```

The first parameter is a function to be executed. The second parameter indicates the number of milliseconds before execution.The execution will occur after the given delay. It is the method of window object but it can also be called without window prefix.

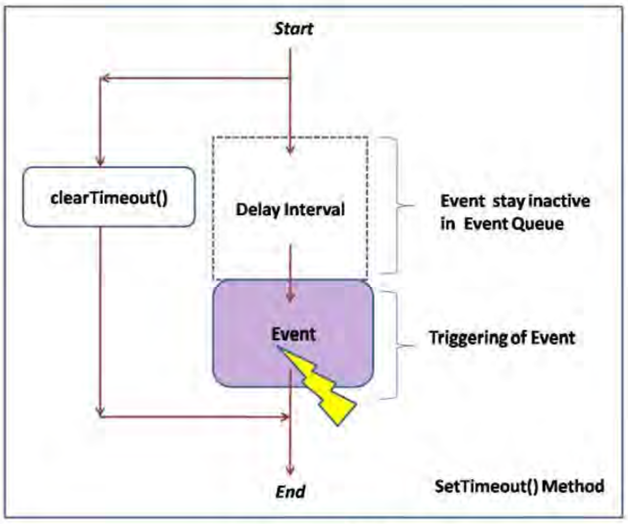


Figure 3.2

Example 3.6 : setTimeout()

```
<html>
<button onclick="setTimeout(myFunction, 1000)">Try it</button>
    <script type="text/javascript">
         function myFunction() {
              alert('Hello'); }
    </script>
```

```
</html>
```

3.8 window.clearTimeout() method

The clearTimeout() method stops the execution of the function specified in setTimeout().

Syntax:

```
window.clearTimeout(timeoutVariable)
```

The window.clearTimeout() method can be written without the window prefix. The clearTimeout() method uses the ID returned from setTimeout():

```
myVar = setTimeout(function, milliseconds);

clearTimeout(myVar);
```

If the function has not already been executed, the execution can be stopped by calling the clearTimeout() method.

Example 3.7: clearTimeout()

```
<html>
<input type="button" onclick="myFunction();" value="setTimeout"/>
    <script type="text/javascript">
        var temp= setTimeout(myFunction,1000);
         function myFunction() {
            alert('Hello'); }
            clearTimeout(var);
    </script>
</html>
```

3.9 setInterval() Method

Because JavaScript does not directly provide a timer, it is possible to use the Window object's setInterval() method to serve the same purpose. The setInterval() method is supported in JavaScript 1.2 and higher. The setInterval() method repeatedly calls a function or evaluates an expression each time a time interval (in milliseconds) has expired. This method continues to execute until the window is destroyed or the clearInterval() method is called. Same as setTimeout(), but repeats the execution of the function continuously at every given time-interval.

Syntax:

```
window.setInterval(function, milliseconds)
```

The window.setInterval() method can be written without the window prefix. The first parameter is the function to be executed. The second parameter indicates the length of the time-interval between each execution.

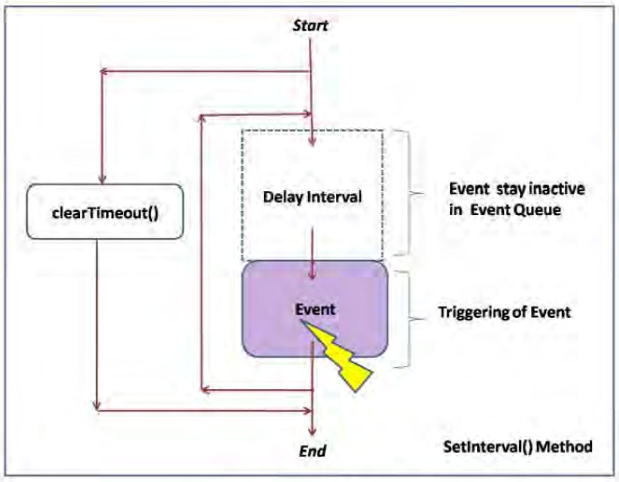


Figure 3.3

Example 3.8: setTimeout()

```
<html>
<button onclick="setInterval(myFunction, 1000)">Try it</button>
    <script type="text/javascript">
        function myFunction() {
            alert('Hello'); }
    </script>
</html>
```

3.10 window.clearInterval() method

The clearInterval() method stops the executions of the function specified in the setInterval() method.

```
window.clearInterval(timerVariable)
```

The window.clearInterval() method can be written without the window prefix. The clearInterval() method uses the variable returned from setInterval():

```
myVar = setInterval(function, milliseconds);

clearInterval(myVar);
```

Example 3.9: clearInterval()

```
<html>
<input type="button" onclick="myFunction();" value="setTimeout"/>
    <script type="text/javascript">
        var temp= setInterval(myFunction,1000);
```

```
            function myFunction() {
                alert('Hello'); }
                clearInterval(var);
        </script>
    </html>
```

3.11 Self Learning Exercise

Q.1 Which dialog box is used for having input from the user-

a) Alert box

b) Confirm box

c) Prompt box

d) None of the above

Q.2 .... is a built - in JavaScript function which can be used to execute another function after a given time interval.

a) Timeout( )

b) TimeInterval( )

c) setTimeout ( )

d) All of the above

Q.3 The different types of dialog boxes supported by JavaScript are-

i) Alert dialog box

ii) Information dialog box

iii) Confirm dialog box

iv) Prompt dialog box

a) i, ii and iii only

b) i, iii and iv only

c) i, ii and iv only

d) All i, ii, iii, and iv

3.12 Summary

- Messaging is an important tool for communication of JavaScript with the user.
- alert box, confirm box and prompt box are three special types of dialog boxes which can commonly be used in JavaScript.
- Alert boxes are used as a one-way messaging systems where JavaScript can show his message to user but user can not interact with it.
- In confirm dialog box the user can respond to this message by clicking on ok or cancel buttons on it. This is also called confirmation box.
- The prompt box is a dialog box with a text box to enter some value.
- setTimeout(), setInterval(), clearTimeout(), and clearInterval() are four special time event methods of window object

3.13 Glossary

Dialog box - a box which is used to prompt a user. There are three dialog boxes- alert box, prompt box and confirm box.

Event queue – a queue which holds waiting for events.

3.14 Answers to Self-Learning Exercise

Q.1 (c)

Q.2 (a)

Q.3 (a)

3.15 Exercise

Q. 1 Write script in the JavaScript to display current time.

Q. 2 Write script to add two numbers. Read these two numbers using input dialog box.

3.16 Answers of Exercise

Q1.

```
<html>
     <script type="text/javascript">
          setInterval(currentTime, 1000);
          function currentTime() {
               var d = new Date();
               var x="Current Time "+d.getHours( )+ ":" + d.getMinutes() + ":"
                    + d.getSeconds();
               alert(x);
     </script>
<body onload=currentTime();></body>
</html>
```

Q2.

```
<!DOCTYPE html>
<html>
     <script type="text/javascript">
          var num1=prompt("Enter Num 1 : ","Num 1");
          var num2=prompt("Enter Num 2 : ", "Num 2");
          num1=parseInt(num1);
          num2=parseInt(num2);
          alert(num1+num2);
     </script>

</html>
```

References and Suggested Readings

1. JavaScript, A Beginner's Guide, Third Edition by John Pollock, O'Reilly Media, March 2014.
2. Head First JavaScript Programming A Brain-Friendly Guide by Eric T. Freeman, Elisabeth Robson, Tata McGraw Hill, 2008.
3. JavaScript: A Beginner's Guide by John Pollock, McGraw-Hill/Osborne, 2004.
4. JavaScript Tutorial - W3Schools www.w3schools.com/js/

Unit-4
XML and AJAX

Structure of the Unit

4.1 Introduction

XML stands for "*eXtension Mark-up Language*". XML *is a text-based mark-up language derived from Standard Generalized Mark-up Language* (SGML).

XML is a:

➢ Meta-language. (A meta-language is a language that is used to define other language)

- Smaller version of SGML. (It is easy to master as compared to SGML, which is a very complex meta-language)

Major advantages of XML are:

- You can define data structure.
- You can make these structures platform independent.
- Process XML defined data automatically.
- Define your own tags.
- XML was designed to both human and machine readable.
- XML is excellent for long-term.

4.2 XML – Declarations

Basic Tags of XML

In XML, each individual piece of information is "marked up" (a marker shows the meaning to the associated data) with a tag that attaches meaning to the information. The unit of data to which a meaning has been attached is called an "element". An 'element' consists of 'start tag', 'content' and 'end tag'.

-

<name> Poornima </name>

Here in this element, <***name***> is start tag, ***Poornima*** is content, <***/name***> is end tag.

When required, an "attribute" can be derived in the start tag of element, allowing more detailed information to be assigned to the data.

For example-

<name ID = "A01001" > Poornima </name>

Here, ***name*** tag has attribute ***ID*** which has value ***A01001*** and ***Poornima*** is the value of <***name***> tag.

If an element has no content then it is called ***Empty*** tag.

For example-

<name > </name>

Here, ***name*** tag is an empty tag.

XML Tags Rules

There are some rules to write tags in XML. They are:

1) XML tags are case-sensitive.

   In XML, start and end tags should be written in same case, i.e. small case, upper case or mixed. For example-

   <name > Sunil </Name>

   Here, this syntax will lead to an error because difference in case of start tag ***<name*** > and end tag ***</Name>*** (Notice the 'N' in place of 'n'). It must be written as:
   <name > Sunil </name>

2) XML tags must be closed in an appropriate order.

   An XML tag opened inside another element must be closed before the outer element is closed. For example:

   <name >

   <first_name> Sunil </first_name>

<last_name> Sharma </last_name>

</name >

Here, ***<name >*** is outer tag and ***<first_name>*** and ***<last_name>*** are inner tags.

XML – Declaration

XML declaration contains details that prepare an XML processor to parse the XML document. It is optional, but when used, it must appear in first line of the XML document.

Following is the syntax of XML declaration:

```
<?XML
version= "version_ no"
        encoding = "encoding_declaration"
        standalone= "standalone_status"
        ?>
```

Here ***version, encoding*** and ***standalone*** are the three parameters of XML declaration.

Following table shows the possible values of each parameter and their meaning:

Parameter Name	Parameter Value	Description
Version	1.0	Tells the version of the XML standard used.

Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	Defines the character encoding used. Default is UTF-8.
Standalone	Yes, No	Yes: Tells the parser that XML document has no external DTD (i.e. there is an internal DTD). No: Tells the parser that XML document has an external DTD. Default is No.

Table 4.1: XML declaration parameters & their values

XML Declaration rules

Following are rules to use XML declaration:

1) XML declaration must be written at the first position of the first line in the XML document, if used and must have version number.
2) If the XML declaration is included, it must contain version number.
3) Parameter names and values are case-sensitive.
4) Names must be in lower case.
5) Order of placing the parameters must be followed as version, encoding and standalone.
6) XML declaration has ?> instead of closing tag </?xml>.

Basic example of XML

With XML tags you can define the types of data but often data is more complex and it can consist of several parts.

```
<?XML version= "1.0" encoding = "UTF-8"?>
<Car>
<Brand> Volvo </Brand>
<Type> V40 </Type>
<Colour> green </Colour>
</Car>
```

In this example ***<XML version= "1.0" encoding = "UTF-8"?>*** is very essential to represent the version of XML. In editors this part of code is generated by default. Next line ***<Car>*** is starting tag of script. ***<Brand>*** is child tag of ***<Car>*** tag. ***<Brand>*** tag represents the brand name of car, which is ***Volvo*** in this example. Similarly ***<Type>*** and ***<Colour>*** tags has data values ***V40*** and ***green*** respectively. Finally the ***</Car>*** is ending tag of XML.

4.3 Root Element, Child Elements

Root Element

In any mark-up language, the first element to appear is called the ***root element*** which defines what kind of document the file will be. All XML documents must have one root element. The root element is working as cabinet which keeps all files together. The root element encapsulates all other elements, so root element is also known as ***parent element***. Here is an example of a XML document with the root element 'phonebook'-

```
<phonebook>
        <number> 9123456789 </number>
        <number> 9123456789 </number>
</phonebook>
```

Here, root element ***phonebook*** surrounds the other element ***number*** in XML file.

Child Element

In XML, an element can have many or no child element. An element defined just below root element is called child element of root element. Element ***b*** is the child of element ***a***, when ***b*** is contained within element ***a,*** and is exactly one level below from element ***b***. In other words, ***a*** is parent of ***b***.

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

Here, element ***a*** is parent element of element ***b*** and ***b*** is parent element of element ***c***.

For example:

Let's consider another example of our vicinity:

```
<country>
```

```
<state>
<district>
<area>  200 km2  </area>
<climate>  sunny  </climate>
</district>
/<state>
</country>
```

Here, element state and district are children of parent element country.

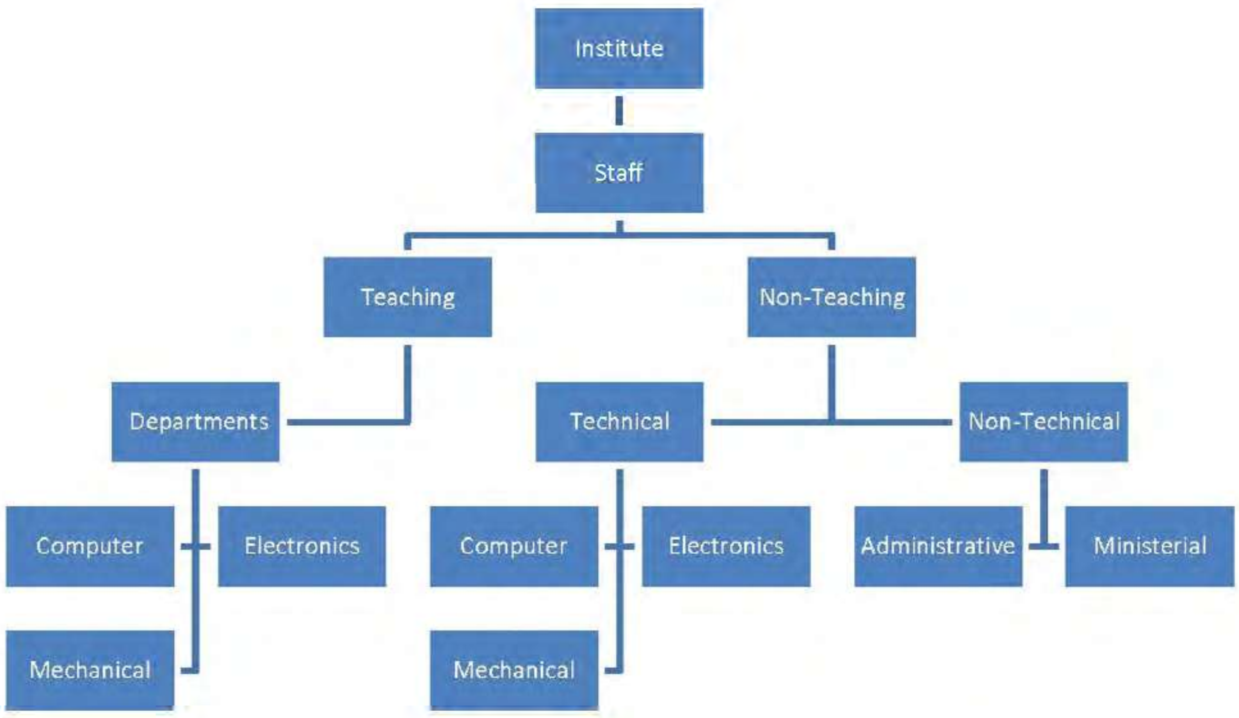


Figure. 4.1: Example of XML tree for technical institute

XML tree in Figure. 4.1 shows the parent child relationship in a technical institute. In this example the *Institute* is root element or parent element. *Staff* is child element of *Institute* and it has further two child elements *Teaching* and *Non-Teaching*. *Teaching* has one child element *Departments* which have further three child elements: *Computer*, *Electronics*, and *Mechanical*. Similarly *Non-Teaching* has two child elements *Technical* and *Non-Technical*. *Technical* is having three child elements: *Computer*, *Electronics*, and *Mechanical* and *Non-Technical* is having *Administrative* and *Ministerial* as child elements.

4.4 Element Attributes, Entity References, Comments

XML Element Attributes

In XML document attributes are part of an XML element and describe the properties of an element. Attribute gives more information about XML elements and there can be more than one attributes for an element. These are simple type definitions that cannot contain other element. Attributes can also be assigned an optional default value. Syntax for writing attributes is as follows:

```
<element-name    attribute1  attribute2 >
        ...... content .....
</element-name>
```

Here, ***attribute1*** and ***attribute2*** has following form:

name = "value"

In XML document attribute values must always be quoted either single or double.

For example:

```
<institute >
```

```
        <courses>
                < course offered  = "Computer Science" >
                < course offered  = "Electronics Engg." >
</courses>
   </ institute >
```

There are some constraints for the attributes:

1) Attributes cannot have multiple values but elements can.
2) Attributes cannot contain tree structure but elements can.
3) Attributes are not easily expandable.

Types of Attributes

Following are types of attributes in XML:-

StringType

StringType declares any literal string as a value. CDATA is StringType and known as character data. This means, any string of non-markup character is a legal part of the attribute.

TokenizedType

This is more constrained type. The validity constraints noted in the grammar are applied after the attribute value is normalized. The TokenizedType attributes are given as:-

- **ID:** It is used to specify the element as unique.
- **IDREF:** It is used to reference an ID that has been named for another element.
- **IDREFS:** It is used to reference all IDs of an element.
- **ENTITY:** It indicates that the attribute will represent an external entity in the document.

- **ENTITIES:** It indicates that the attribute will represent external entities in the document.
- **NMTOKEN:** It is similar to CDATA with restrictions on what data can be part of the attribute.
- **NMTOKENS:** It is similar to CDATA with restrictions on what data can be part of the attribute.

EnumeratedType

This has a list of predefined values in its declaration. Out of which, it must assign one value. There are two types of enumerated attribute:

- **NotationType:** It declares that an element will be referenced to a NOTATION declared somewhere else in the XML document.
- **Enumeration:** Enumeration allows you to define a specific list of values that the attribute value must match.

Entity References

An entity reference is an alternative name for a series of characters. One can use an entity in the &name; format, where name is the name of the entity. There are various predefined entities in XML; furthermore one can declare entities in a DTO (Document Type Definition).

Following are the ways in XML to reference an entity:

1) A numeric character reference refers to a character.

   A numeric character reference can be specified in two formats: ***&#nnnn;*** or ***&#xhhhh***,

   where ***n*** decimal digits or ***h*** hexadecimal characters identify the Unicode character code of the referenced character.

2) An entity reference refers to a series of characters.

Some characters have a special meaning in XML. If you place a character like "<" inside an XML element, it will generate an error.

Let us consider an example which will explain the need of reference in XML document-

```
< message >   salary < 1000   < /message >
```

Here, "<" (less than symbol in ***salary* < 1000**) will generate an error. To overcome this error we have to replace it with entity reference as shown below:

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

Following table shows predefined entity references available in XML:

Symbol	Entity Reference	Meaning
<	&lt;	Less than
>	&gt;	Greater than
&	&amp;	Ampersand
'	&apos;	Apostrophe
"	&quot;	Quotation mark

Table 4.2: Predefined entity references in XML

For example:

```
<entity check>
        <predefined> 2 &lt; 5 </predefined>
</entity check>
```

```
<entity test>
        <predefined> 2 < 5 </predefined>
</entity test>
```

Here, ***<entity check>*** is root element of ***<predefined>*** and ***<declared>*** tags. In tag ***<predefined>*** we want to compare 2 and 5, but cannot write less than (<) symbol in XML language. So we have written &lt; in the place of less than (<) symbol.

Comments in XML

XML comments are generally used for understanding purpose of an XML code. Comments are similar to HTML comments. These can be used to include related links, information and terms. They are visible only in the source code; not in XML code. Comments may appear anywhere in XML code.

XML comment has following syntax:

```
< !-- comments -->
```

XML comments starts with ***<!--*** and ends with ***-->***. ***Y***ou can add textual notes as comments between the characters.

Let us consider an example to describe the XML comments.

```
<?XML version ="1.0" encoding = ISO-8859-15
<!-- Students ranks of final year -->
<class_ list>
        <student>
        <name> Krishna </name>
        <rank> 1 </rank>
</student>
<student>
        <name> Ram </name>
        <rank> 2 </rank>
```

```
</student>
</class_ list>
```

Here, in example student ranks are displayed. Second line ***<!-- Students ranks of final year -->*** is a single line comment. XML parser does not read this comment line.

One can use XML comments to temporarily remove code from XML document. For example in above XML code if we want to temporarily remove student ***Krishna*** then this code will look like:

```
<?XML version ="1.0" encoding = ISO-8859-15
<!-- Students ranks of final year -->
<class_ list>
        <!--
                <student>
                <name> Krishna </name>
                <rank> 1 </rank>
                </student>

        -->
        <student>
                <name> Ram </name>
                <rank> 2 </rank>
        </student>
        </class_ list>
```

Here, in example student ranks are again displayed but record for student ***Krishna*** is made comment and it is a multiline comment.

Following constraints are needed to be applied to XML comments:

- Comments cannot appear before XML declaration.
- Comments may appear anywhere (after XML declaration) in a document.
- Comments must not appear within attributes value.
- Comments cannot be nested inside the other comments.

4.5 AJAX - XMLHttpRequest Object

AJAX means Asynchronous JavaScript And XML. It is not a programming language. AJAX uses browser built-in XMLHttpRequest object in combination with JavaScript and HTML DOM. The XMLHttpRequest object is the key to AJAX. This has been available ever since internet explorer 5.5. But all browsers support the XMLHttpRequest object.

The XMLHttpRequest object is used to exchange data with a server behind the scenes. That's how it is possible to update parts of web page, without reloading the whole page. XMLHttpRequest (XHR) is an API that can be used by JavaScript JScript, VBscript, and other web browser scripting languages to transfer and manipulate XML data to and from a web server using HTTP, establishing an independent connection channel between a webpage's client-side and server-side.

4.6 Sending AJAX request

To send a request to a server, we use the open( ) and send( ) methods of the XMLHttpRequest object. Let us consider few methods of XMLHttpRequest:

- **xhttp.open("method", url, async);**

  Here,

  method is Type of request which can be GET or POST,

  url is Server or file location, and

async is True (for asynchronous) or false (for synchronous)

By this method we open the server request. We can send the data by get or post method. For example:-

xhttp.open("GET", "ajax_info.txt", true);

- **xhttp.send( );**

  This method sends the request to the server and used only for GET method. For example:-

  xhttp.send( );

- **xhttp.send(String);**

  This method also sends the request to the server but in the form of string and used only for POST method.

- **getallresponseheader();**

  This method returns the complete set of HTTP Headers as a string and gives the response to user that connection has been established.

- **getresponseheader(headername);**

  This method returns the value of the specified HTTP header. As we know Httpheader confirm the connection establishment. This method returns the value of HTTP header.

  XMLHttpRequest properties:-

onreadystatechange:- An event handler for an event that fires at every state change.

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:

Table 4.2: 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

4.7 Handling AJAX responses

The readystate property holds the status of the XMLHttpRequest. The onreadystatechange property defines a function to be executed when the ready state changes. The status property and the statustext property hold the status of the XMLHttpRequest object.

Table 4.3: readystate properties and their description.

Property of response	Description
onreadystatechange	Defines a function to be called when the ready state property change
readystate	Holds the status of the XMLHttp as- 0: requests not initialized 1: sever connection established 2: request received 3: processing request 4: request finished and response is ready

Status	200: "ok" 403: "Forbidden" 404: "Page not found" for complete list please go to Http Message Reference
statustext	returns the status-text (e.g. "ok" or "not found")

Let us consider an example of Request-Response when readystate is 4 and status is 200 the response is ready:-

```
function loadDoc()
{
Var Xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function()
{
If (this.readystate = 4 && this.Status ==200)
{
    Document.getElementId("demo").innerHTML
    = This.responseText;
        }
};
xhttp.open ("GET", "ajax_info.txt", true);
xhttp.send();
}
```

Above is a basic request – response example. First we make a variable xhttp by creating XMLHttpRequest. This variable creates an object which transfers from function. If the readystate is 4 state is 200 then, server upload this document in the server. After that xhttp.open()function AJAX text file upload to sever.

4.8 learn VII – Adding AJAX Functionality in JavaScript

AJAX is an efficient way for a web application to handle user interactions with JavaScript. AJAX interactions are initiated by JavaScript code. When the AJAX interaction is complete, JavaScript updates the HTML source of the page. These changes are made immediately without requiring a page refresh. Dynamically updation of data on a page and submit partial forms from the page.

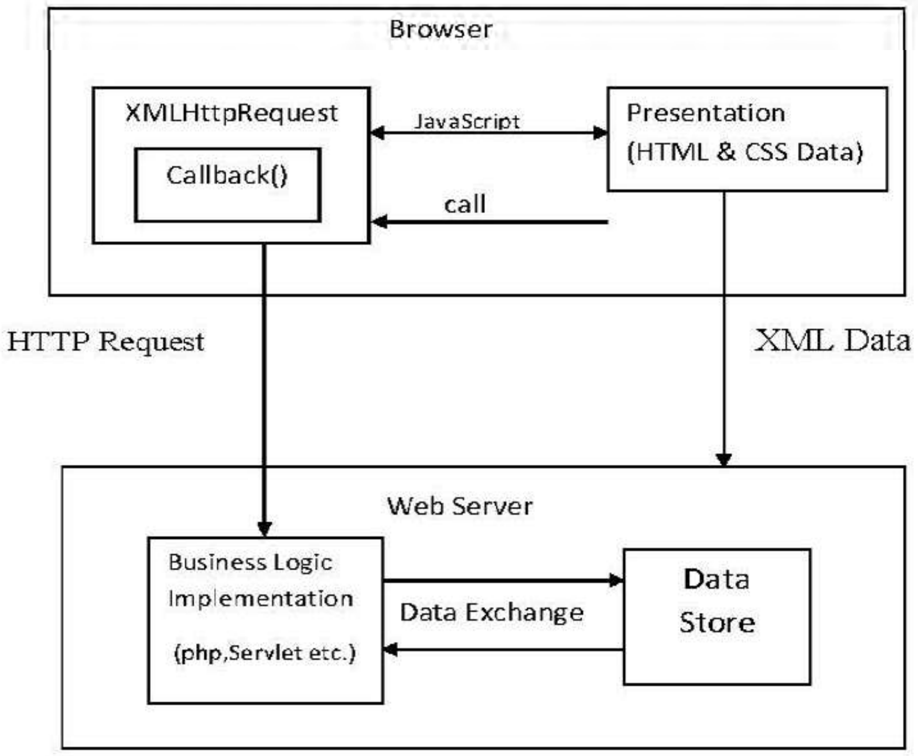


Figure. 4.2: Block diagram of Interface of JavaScript and web page through AJAX.

Following are the steps (as shown in Figure. 4.2) to fetch data using AJAX:-

- The used triggers are events, for example by pressing a key when typing in a name. Firstly JavaScript call to a function that initializes an XMLHttpRequest object.

- This XMLHttpRequest object is configured with a request parameter that includes the ID of the component that triggered the event and any value that the user entered. This XMLHttpRequest object then makes an asynchronous request to the web server.

- On the web server, an object such as a Servlet or Listener handles the request. Data is retrieved from database and response going towards XML documents.

Finally XMLHttpRequest object receiving the XML data using a call back function processes it and updated the HTML DOM (Document Object Model) to display the page containing the new data.

Let us consider an example of AJAX and JavaScript with AJAX:-

```
<Script >
$ (document).ready (function(){
$ (“button”).click (function(){
$ ajax (url : “demo_test.txt”, success: function(result){
                                    $(“#div1”).html(result);
                  }});
</Script >
</head>
<body>
<div id =” div1”><{h2>Let JQuery AJAX change this Text</h2> </div>
<button> Get External Content </button>
```

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

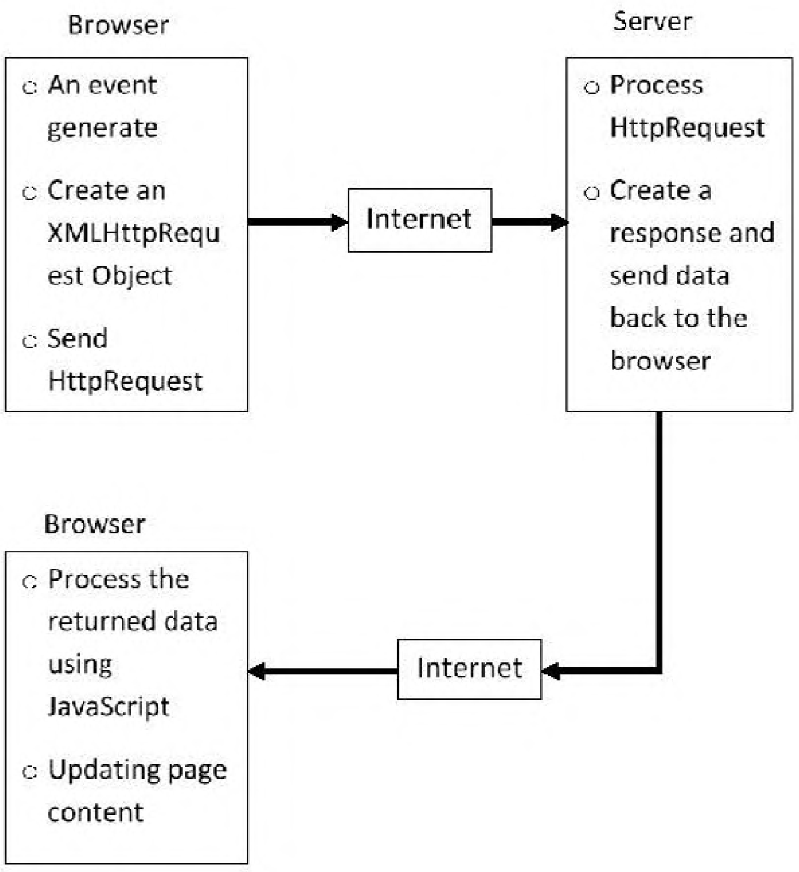
Output will look like:

Let JQuery AJAX change this Text

Get External Content

4.9 learn VIIII – Adding AJAX Functionality to a Web

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



Browser
An event generate
Create an XMLHttpRequest Object
Send HttpRequest
Internet
Server
Process HttpRequest
Create a response and send data back to the browser
Internet
Browser
Process the returned data using JavaScript
Updating page content

Figure. 4.3: How the AJAX works into the web page

We can describe this by following steps:-

- An event occurs in a web page (the page is loaded, a button is clicked).
- An XMLHttpRequest object is created by JavaScript.
- The XMLHttpRequest object sends a request to a web server.
- The server process the request.
- The server sends a response back to the web page.
- The response is read by JavaScript.
- Proper action (like page update) is performed by JavaScript.

4.10 Self learning exercise

Q.1 How HTML is different from XML?

Q.2 What are the benefits of XML?

Q.3 Define attribute in an XML Document.

Q.4 What is XML DOM Document?

Q.5 State True or False:-

a) An attribute provides more or additional information about an element than otherwise.
b) Can we have empty XML tags?
c) Can, I execute a XML code?
d) Can, I replace HTML with XML?
e) Is XML is case sensitive?

Q.6 what are differences between AJAX and JavaScript?

4.11 Summary

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet. By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Sun Microsystems engineer Jon Bosak developed a charter and recruited collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web. It is usually stored in a simple text file and is processed by special software that is capable of interpreting XML.

AJAX is a framework that leads Web programming into the new generation of using and developing the internet. AJAX is a new implementation of established web development technologies to gain interactivity between users and servers through multiple client sided server connections in the background. It allows integrating web applications in browsers; as a consequence the borders to desktop applications are disappearing.

4.12 Glossary

SGML- *Standard Generalized Mark-up Language*
XML- *eXtension Mark-up Language*
HTML- *HyperText Mark-up Language*
AJAX- *Asynchronous JavaScript And XML attribute*
Element- *The unit of data to which a meaning has been attached*
Empty tag- *element without any content value*

Browser- *Software used to navigate the Internet. Google Chrome, Firefox, Netscape Navigator and Microsoft Internet Explorer are few examples.*

JavaScript- *A programming language used almost exclusively to manipulate content on a web page.*

4.13 Answers to self-learning exercise

Ans. 1) Difference between HTML and XML:-

HTML	XML
HTML is an abbreviation for HyperText Markup Language.	XML stands for eXtensible Markup Language.
HTML was designed to display data with focus on how data looks.	XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
HTML is a markup language itself.	XML provides a framework for defining markup languages.
HTML is a presentation language.	XML is neither a programming language nor a presentation language.
HTML is case insensitive.	XML is case sensitive.
HTML is used for designing a web-page to be rendered on the client side.	XML is used basically to transport data between the application and the database.
HTML has its own predefined tags.	While what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the XML document.

Ans. 2)

Following are the benefits of XML:-

- Simplicity - Information coded in XML is easy to read and understand, plus it can be processed easily by computers.
- Openness - XML is a W3C standard, endorsed by software industry market leaders.
- Extensibility - There is no fixed set of tags. New tags can be created as they are needed.
- Self-description - XML documents can be stored without schemas because they contain meta-data; any XML tag can possess an unlimited number of attributes such as author or version.
- Contains machine readable context information- Tags, attributes and element structure provide context information opening up new possibilities for highly efficient search engines, intelligent data mining, agents, etc.
- Separates content from presentation - XML tags describe meaning not presentation. The look and feel of an XML document can be controlled by XSL style sheets, allowing the look of a document (or of a complete Web site) to be changed without touching the content of the document. Multiple views or presentations of the same content are easily rendered.
- Supports multilingual documents and Unicode - This is important for the internationalization of applications.
- Facilitates the comparison and aggregation of data - The tree structure of XML documents allows documents to be compared and aggregated efficiently element by element.

- XML simplifies data transport - One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet. Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications.

Ans. 3) XML elements can have attributes. By the use of attributes we can add the information about the element. XML attributes enhance the properties of the elements. XML attributes must always be quoted. We can use single or double quote. Let us take an example of a book publisher.

< book publisher = "Tata McGraw Hill"> < / book >
Here, book is the element and publisher is the attribute.

Ans. 4) The DOM defines a standard for accessing and manipulating documents: *"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."*
The HTML DOM defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure. Understanding the DOM is a must for anyone working with HTML or XML. All XML elements can be accessed through the XML DOM. In other words: The XML DOM is a standard for how to get, change, add, or delete XML elements.

Ans. 5)

(a) True
(b) True
(c) False
(d) False
(e) True

Ans. 6)

Differences between AJAX and JavaScript:-

AJAX	JavaScript
AJAX is the technology which uses Server side.	JavaScript is the client side scripting language
AJAX is technology is not language	JavaScript is programming Language to complete task
AJAX does not install Trojan in computer	JavaScript can install Trojan in computer.

4.14 Exercise

Q.1 What is SGML? Write short note on SGML.

Q.2 What are real web application of AJAX currently running in the market?

Q.3 What is AJAX? Explain AJAX application in web development.

Q.4 What is XQuery?

4.15 Answer to exercise

Ans. 1) SGML stands for *Standard Generalized Mark-up Language*. SGML is a standard for defining generalized markup languages for documents. Generalized markup is based on two postulates:-

- Markup should be declarative: it should describe a document's structure and other attributes, rather than specify the processing to be performed on it. Declarative markup is less likely to conflict with unforeseen future processing needs and techniques.

- Markup should be rigorous so that the techniques available for processing rigorously-defined objects like programs and databases can be used for processing documents as well.

HTML was theoretically an example of an SGML-based language until HTML 5, which admits that browsers cannot parse it as SGML (for compatibility reasons) and codifies exactly what they must do instead.

Word processing package and computer typesetting systems commonly use a procedural markup (Helvetica font, 18 points, etc.) which specify how text will be processed or will appear on an output. In contrast, SGML uses a structural markup (example, figure) which enables the description of structured information independent of how the information is processed. Every SGML-based document requires descriptions or a set of rules about structured information in a document. This description is called document type definition (DTD) and the SGML language provides a standard syntax for expressing DTDs. Any information that are marked up or added into an SGML document must follow the descriptions in a DTD. In other words, DTDs defines the specifications for valid components/parts within documents and rules about how subparts are organized. DTDs are explicit forms of preferences about the document that an author has during authoring. Documents of the same type can share the same DTD and each document written is considered as a document instance.

Ans. 2) Facebook, Gmail, Amazon, Twitter etc are applications which are based on AJAX.

Ans. 3) AJAX stands for *Asynchronous JavaScript And XML* and is a technique for creating fast and dynamic web pages. AJAX is about updating parts of a web page, without reloading the whole page. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page. Classic web pages, (which do not use AJAX) must reload the entire page if the content should change. Examples of applications using AJAX: Google Maps, Gmail, Youtube, and Facebook tabs.

The application of ajax is in web development is as following:-

Callbacks: AJAX is used to perform a callback, making a quick round trip to and from the server to retrieve and/or save data without posting the entire page back to the server. By not performing a full postback and sending all form data to the server, network utilization is minimized and quicker operations occur. In sites and locations with restricted bandwidth, this can greatly improve network performance. Most of the time, the data being sent to and from the server is minimal. By using callbacks, the server is not required to process all form elements. By sending only the necessary data, there is limited processing on the server. There is no need to process all form elements, process the ViewState, send images back to the client, or send a full page back to the client.

Making Asynchronous Calls: AJAX allows you to make asynchronous calls to a web server. This allows the client browser to avoid waiting for all data to arrive before allowing the user to act once more.

User-Friendly: Because a page postback is being eliminated, AJAX enabled applications will always be more responsive, faster and more user-friendly.

Increased Speed: The main purpose of AJAX is to improve the speed, performance and usability of a web application. A great example of AJAX is the movie rating feature on Netflix. The user rates a movie and their personal rating for that movie will be saved to their database without waiting for the page to refresh or reload.

These movie ratings are being saved to their database without posting the entire page back to the server.

Ans. 4) XQuery (XML Query) is a query and functional programming language that queries and transforms collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (JSON, binary, etc.). The language is developed by the XML Query working group of the W3C. "The mission of the XML Query project is to provide flexible query facilities to extract data from real and virtual documents on the World Wide Web, therefore finally providing the needed interaction between the Web world and the database world. Ultimately, collections of XML files will be accessed like databases". Let consider an example of Xquery.

References and Suggested Readings

1. XML: The complete reference by Heather Williamson, Tata McGraw Hill,2008
2. AJAX: The complete reference by Thomas A. Powell, The McGraw Hill,2008
3. http://docs.oracle.com/javaee/6/tutorial/doc
4. https://www.tutorialspoint.com/xml
5. https://www.w3schools.com
6. https://developer.mozilla.org

UNIT-5
Installation: Apache & MySQL

Structure of the Unit

5.0 Objective

The objectives in this unit are:

- To install Apache Web Server, PHP & MySQL on PC.
- Test the installation and configuring Apache server. Showing PHP installation info on home page of our local server.

5.1 Introduction

Apache web server is provided by Apache organization which is open source. You can download source code and compile according your requirements. PHP is scripting language which is used to code server side logic. It has got many plugin to connect other third party programs like Apache web server, MySQL etc. MySQL is Database Management system which is SQL base database. It has many languages bindings and it also provide module for PHP language so that we can connect to our database from PHP code itself.

5.2 Software Prerequisites

In this chapter we shall focus upon the following topics

- Operating System : Ubuntu, Windows or Mac
- Minimum RAM: 512 Mb.
- Storage Space minimum 100Mb.
- Working Internet Connection

5.3 Installing Apache & PHP

To install Apache or other software on Ubuntu we can use default package Management tool Advanced Packaging Tool (APT).

It is use to install, upgrade and remove software packages. To install software we will use apt-get command.

First we will update the list of all repositories from where APT tool download packages by following command.

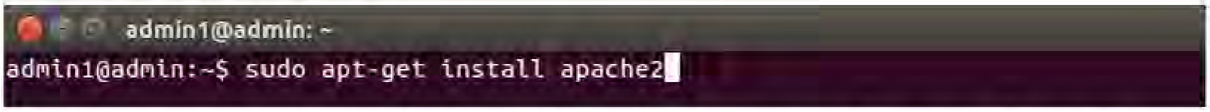
sudo apt-get update

Now we will install Apache package.

Syntax:

sudo apt-get install <Package-name>

sudo apt-get install apache2



```
admin1@admin:~$ sudo apt-get install apache2
```

SUDO is used because to install, upgrade and remove software package we need administrative privilege.

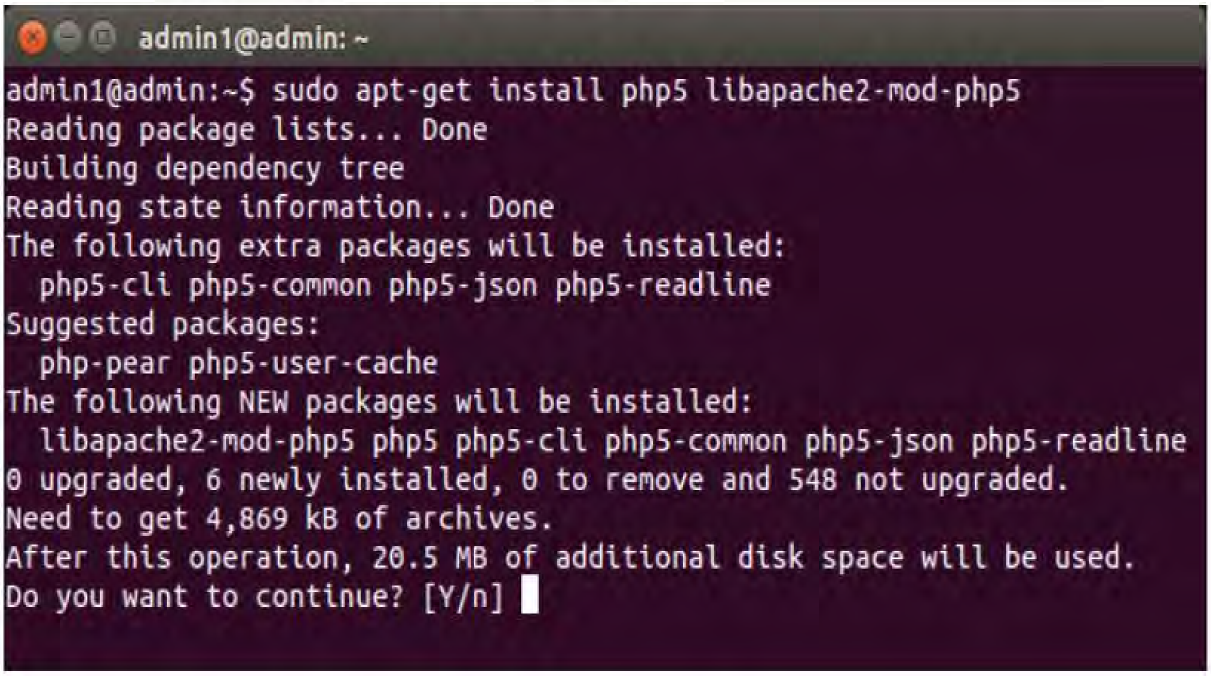
Installing PHP on Ubuntu 14.x.x

To install PHP we will use APT tool. By default in Ubuntu 14.x.x PHP5 is used so we will install PHP5.

In Ubuntu 16.x.x default version of PHP is 7.

To install PHP5

sudo apt-get install php5 libapache2-mod-php5



```
admin1@admin:~$ sudo apt-get install php5 libapache2-mod-php5
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  php5-cli php5-common php5-json php5-readline
Suggested packages:
  php-pear php5-user-cache
The following NEW packages will be installed:
  libapache2-mod-php5 php5 php5-cli php5-common php5-json php5-readline
0 upgraded, 6 newly installed, 0 to remove and 548 not upgraded.
Need to get 4,869 kB of archives.
After this operation, 20.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

5.4 Starting and Testing Apache

Now will configure our Apache web server installation for first use.

We will do following task:

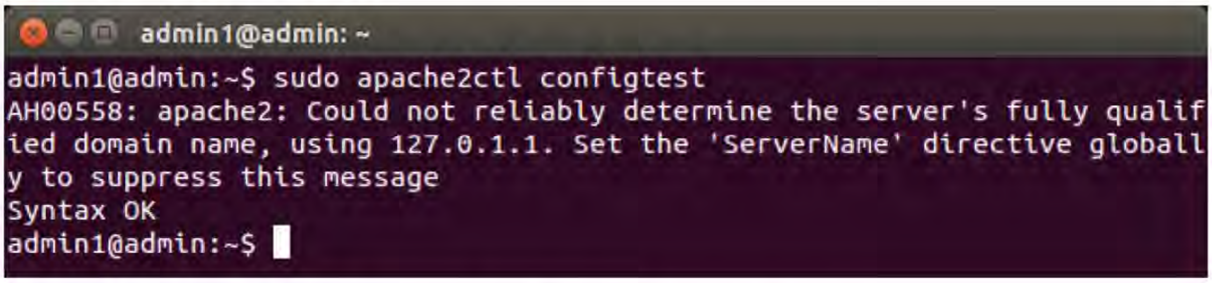
- We will set ServerName
- We will allow Apache Web server in firewall internet connectivity.

All the Apache Web server configuration are stored in apache2.conf file.

It is located in **/etc/apache2/apache2.conf**

To perform Apache configuration test run following command:

sudo apache2ctl configtest



```
admin1@admin: ~
admin1@admin:~$ sudo apache2ctl configtest
AH00558: apache2: Could not reliably determine the server's fully qualif
ied domain name, using 127.0.1.1. Set the 'ServerName' directive globall
y to suppress this message
Syntax OK
admin1@admin:~$
```

Now we can see that it is asking us to set the ServerName.

To edit configuration file we will use gedit which is default text editor in Ubuntu.

To open file in gedit the command used is:

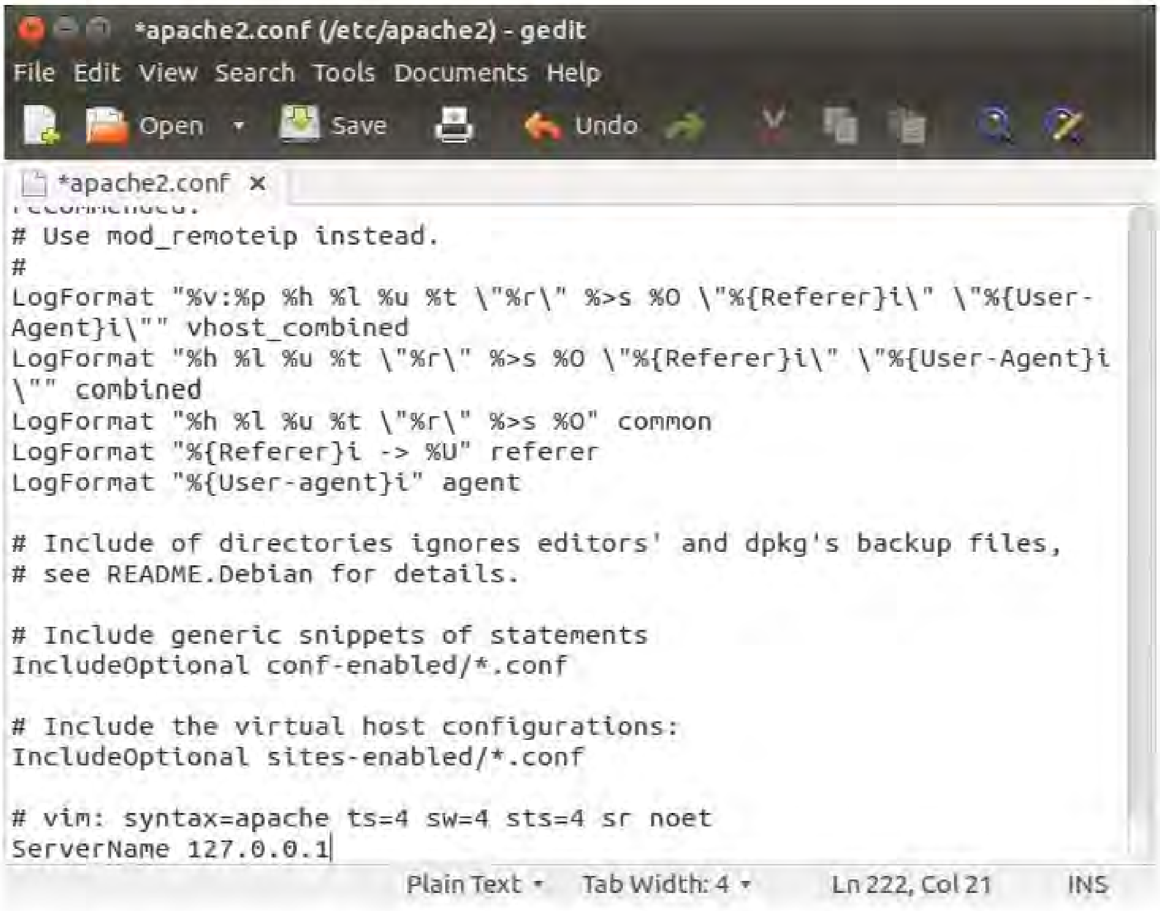
gedit <file-path-including-file-name>

*To edit system file don't forget to use sudo.

sudo gedit /etc/apache2/apache2.conf

Now at the end of file we will add

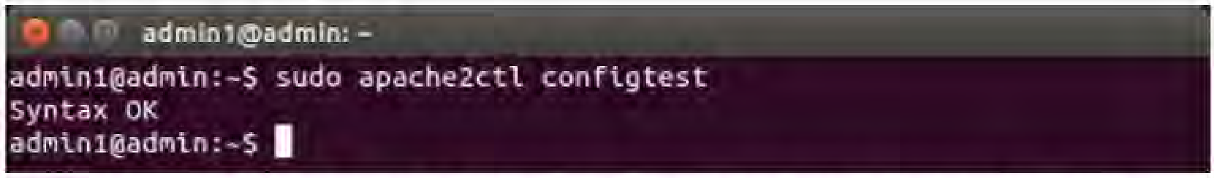
ServerName 127.0.0.1


*apache2.conf (/etc/apache2) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
*apache2.conf
Plain Text
Tab Width: 4
Ln 222, Col 21
INS

```
# Use mod_remoteip instead.
#
LogFormat "%v:%p %h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-
Agent}i\"" vhost_combined
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i
\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

# Include of directories ignores editors' and dpkg's backup files,
# see README.Debian for details.

# Include generic snippets of statements
IncludeOptional conf-enabled/*.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
ServerName 127.0.0.1
```

Save the file and run following command:

sudo apache2ctl configtest

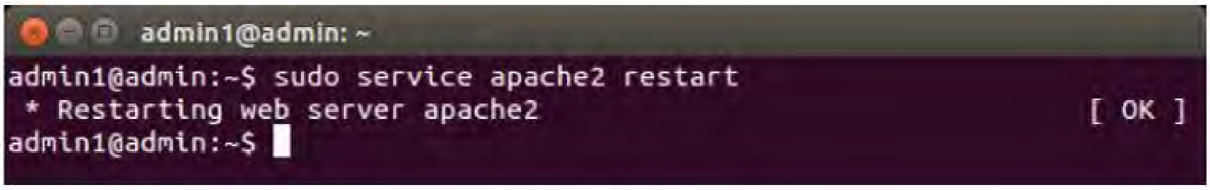


```
admin1@admin:~$ sudo apache2ctl configtest
Syntax OK
admin1@admin:~$
```

Now we have configured the Apache Web Server, we need to restart the server.

Run the following command:

sudo service apache2 restart



```
admin1@admin:~$ sudo service apache2 restart
 * Restarting web server apache2                                    [ OK ]
admin1@admin:~$
```

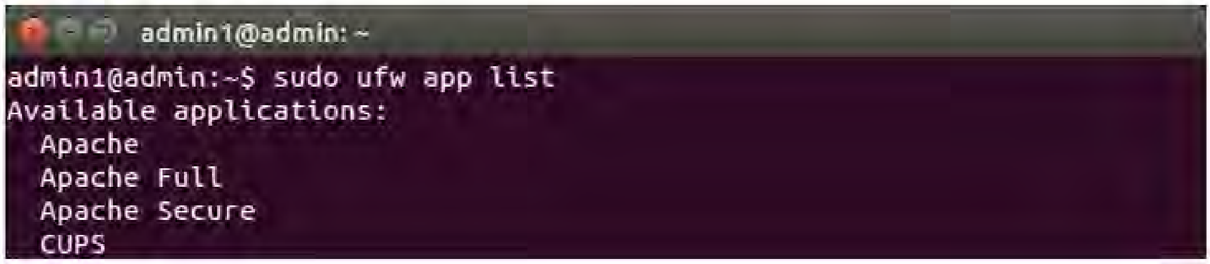
Now we will allow all connection to Apache Web Server in firewall setting.

To do this we will use Uncomplicated Firewall (UFW). UFW is default tool for configuring the firewall in Ubuntu.

Developed to ease iptables firewall configuration, ufw provides a user friendly way to create an IPv4 or IPv6 host-based firewall.

To list all the application which has access to Firewall use following command:

sudo ufw app list

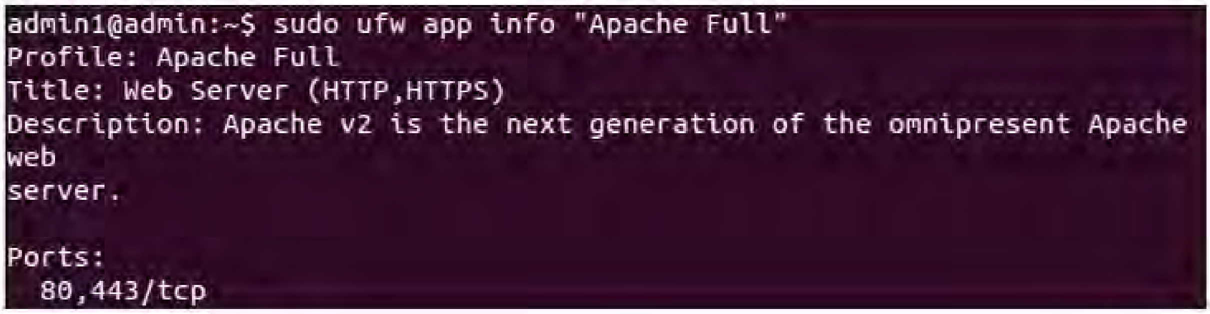


```
admin1@admin:~$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  CUPS
```

Now we can see Apache Full is also there. To list the detail of Apache Full.

We will use following command:

sudo ufw app info "Apache Full"

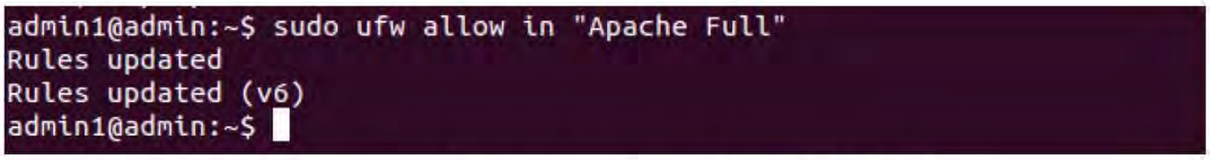


```
admin1@admin:~$ sudo ufw app info "Apache Full"
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache
web
server.

Ports:
  80,443/tcp
```

Now we will allow incoming traffic to it.

To do this we will use following command:

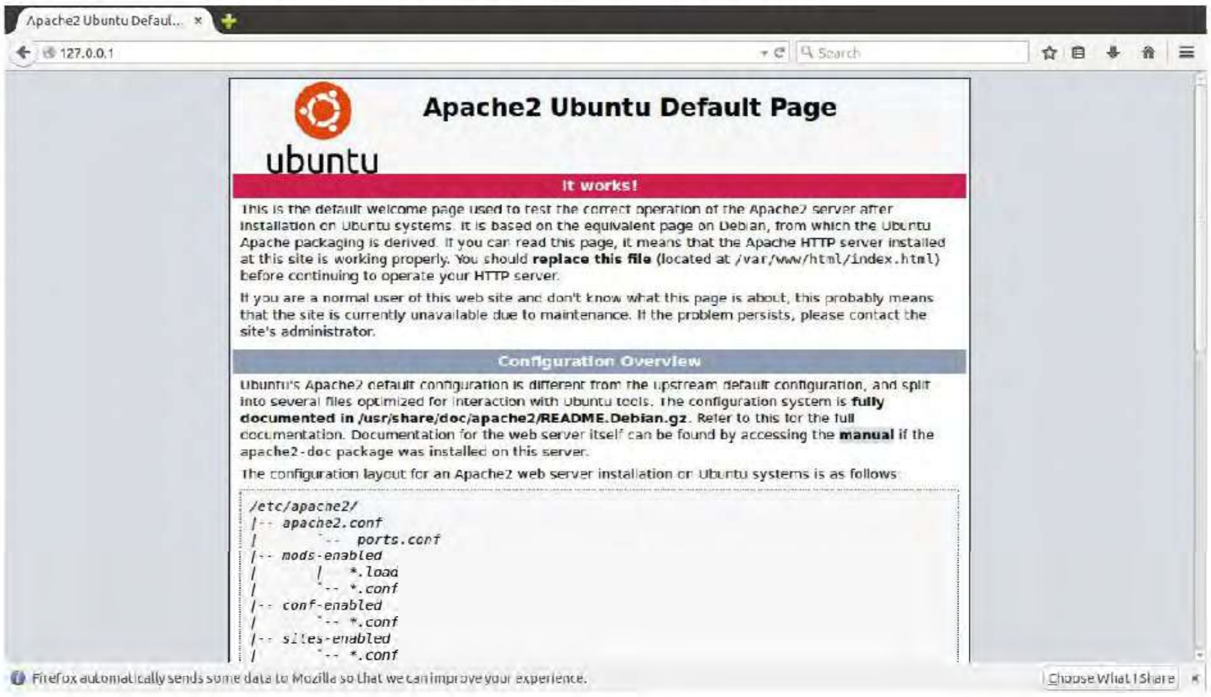
sudo ufw allow in "Apache Full"



```
admin1@admin:~$ sudo ufw allow in "Apache Full"
Rules updated
Rules updated (v6)
admin1@admin:~$
```

Now we are all set. Now open browser and open *http://127.0.0.1* .

You will see the default web page of Apache Web server.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |   *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

Firefox automatically sends some data to Mozilla so that we can improve your experience.　Choose What I Share

5.5 Testing PHP with phpinfo()

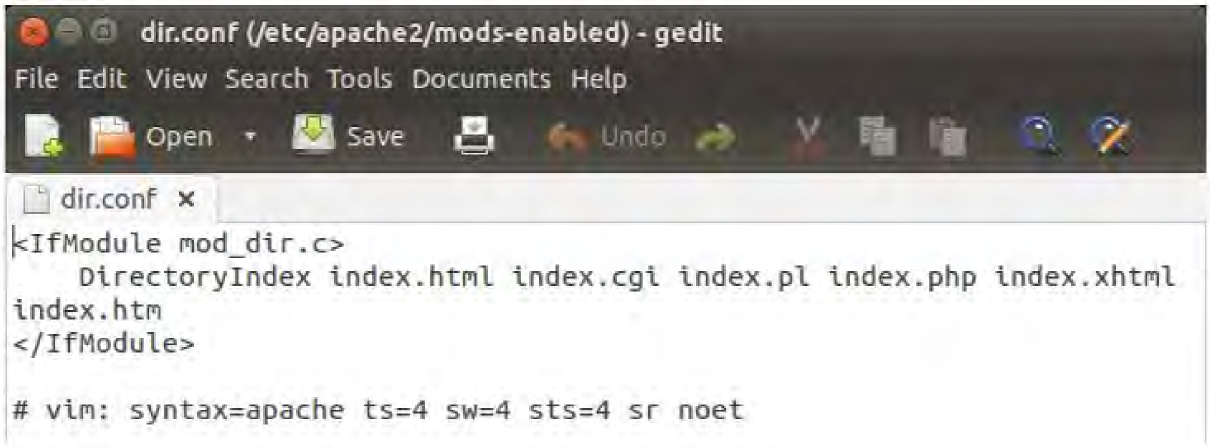
Generally when user request for web page in a directory it looks for index.html by default.

But we want to change it so that it will look for index.php. To do this we will do change 'dir.conf' file which is located in **/etc/apache2/mods-enabled/dir.conf**

To edit we will use gedit tool. To edit the dir.conf file run following command:

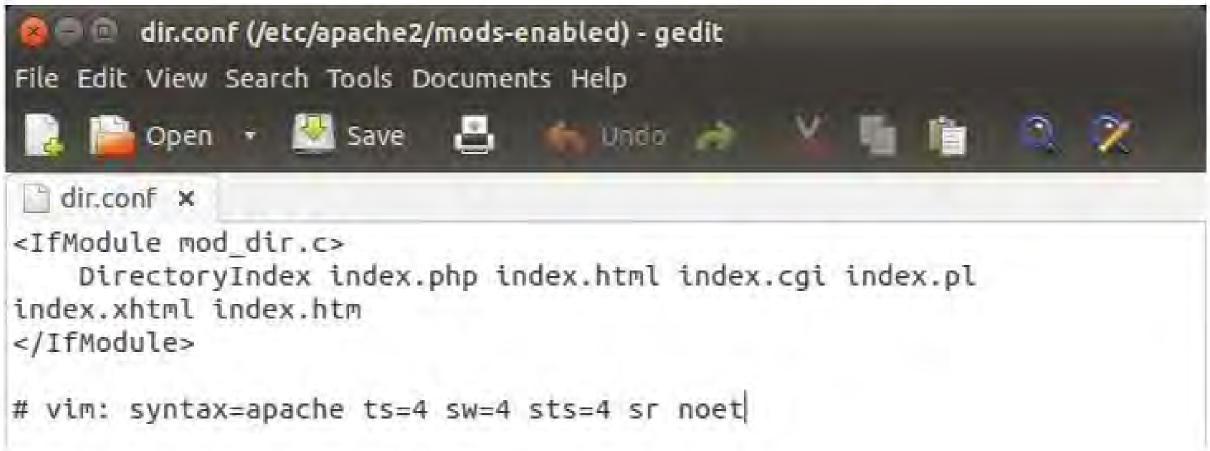
sudo gedit /etc/apache2/mods-enabled/dir.conf

It will have content like:


dir.conf (/etc/apache2/mods-enabled) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
index.htm
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

We will change it and bring index.php before index.html like this:

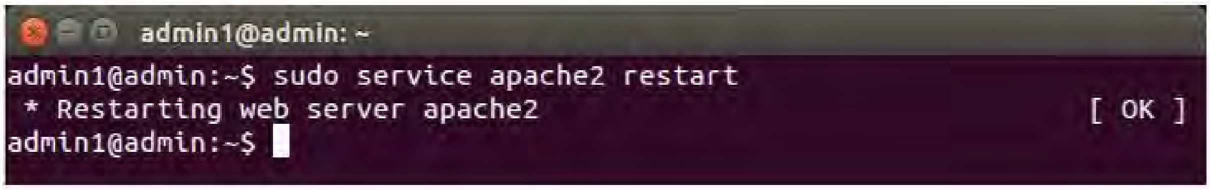

dir.conf (/etc/apache2/mods-enabled) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl
index.xhtml index.htm
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Now To make these changes effective we have to restart the web server.

To do this run following command:

sudo service apache2 restart



```
admin1@admin:~$ sudo service apache2 restart
 * Restarting web server apache2                                   [ OK ]
admin1@admin:~$
```

Now it will look for index.php first instead of index.html

And Apache Web Server serves the files from

/var/www/html/

To add, remove and modify files in this directory we need administrative privilege.

To run file manager with administrative right, run following command:

sudo nautilus

Now on left bar there is computer tab click on it.

You will see list of folder, open **var** folder then open **www** and finally **html**.

This is the directory from where Apache Web Server serves the files.

To test php we will make file named index.php to make file right click select new document then empty document.

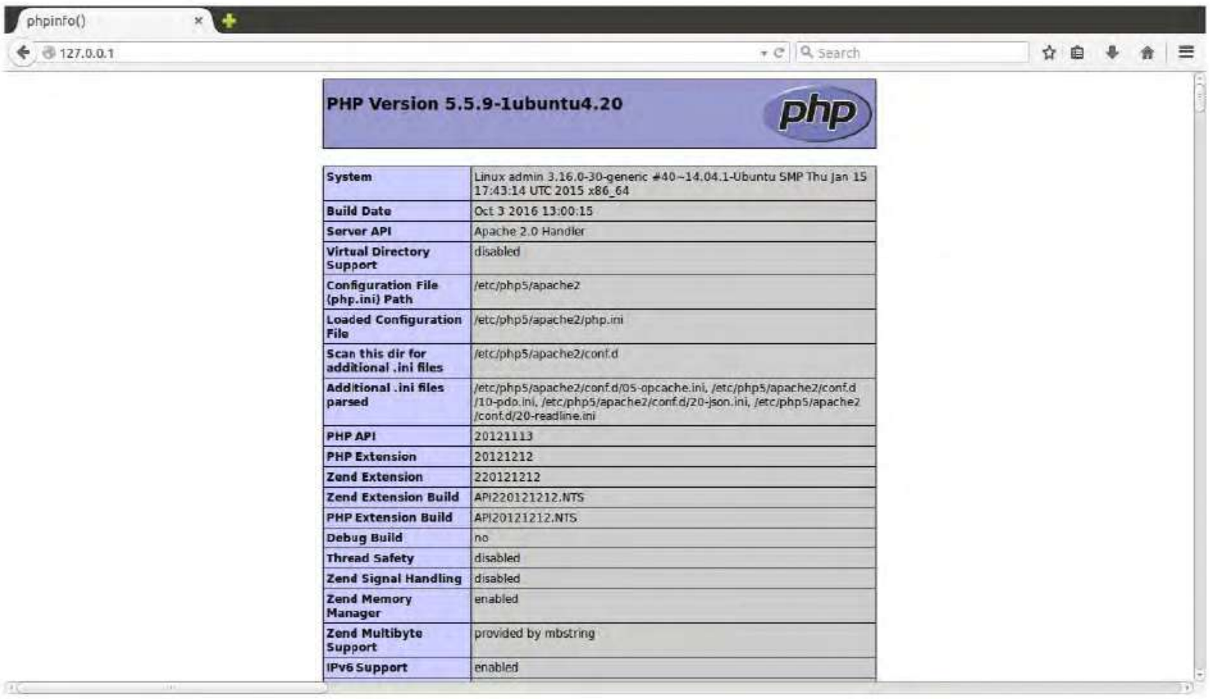
Give it a name index.php.

Open it by double click, add following lines of code:

```
<?php
phpinfo();
?>
```

Now save this file.

Now open web browser and open http://127.0.0.1 . You will see the information of your PHP installation.



PHP Version 5.5.9-1ubuntu4.20

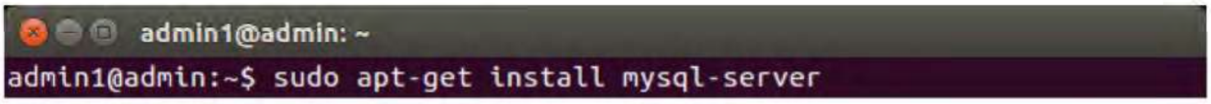
System	Linux admin 3.16.0-30-generic #40~14.04.1-Ubuntu SMP Thu Jan 15 17:43:14 UTC 2015 x86_64
Build Date	Oct 3 2016 13:00:15
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini, /etc/php5/apache2/conf.d/20-readline.ini
PHP API	20121113
PHP Extension	20121212
Zend Extension	220121212
Zend Extension Build	API220121212,NTS
PHP Extension Build	API20121212,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

5.6 Installing MySQL

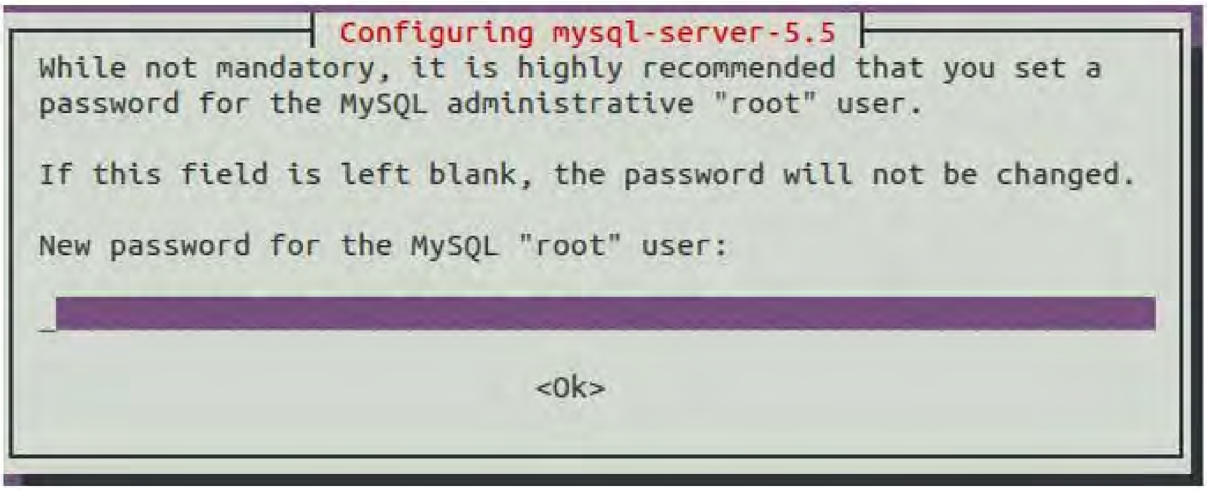
To install MySQL in Ubuntu we will use APT which is a default Package management tool.

To install run following command:

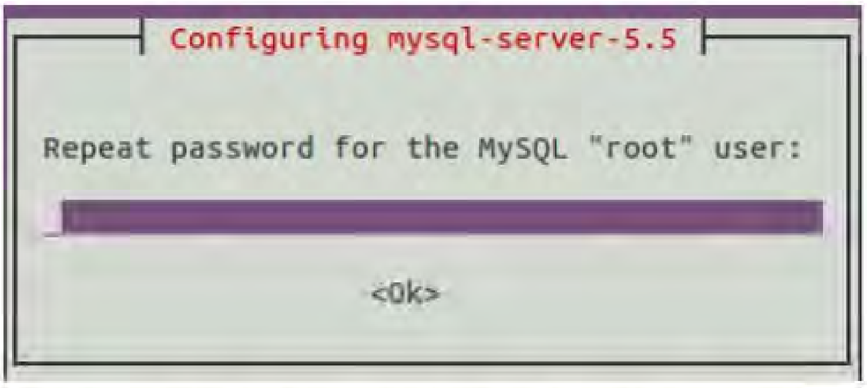
sudo apt-get install mysql-server



```
admin1@admin: ~
admin1@admin:~$ sudo apt-get install mysql-server
```



```
Configuring mysql-server-5.5
While not mandatory, it is highly recommended that you set a
password for the MySQL administrative "root" user.

If this field is left blank, the password will not be changed.

New password for the MySQL "root" user:

                              <Ok>
```

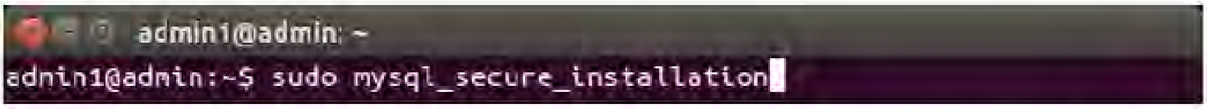


```
Configuring mysql-server-5.5

Repeat password for the MySQL "root" user:

                 <Ok>
```

When prompted set password for root user for the MySQL. When mysql-server get installed we will run some command so that our MySQL server get more secure.

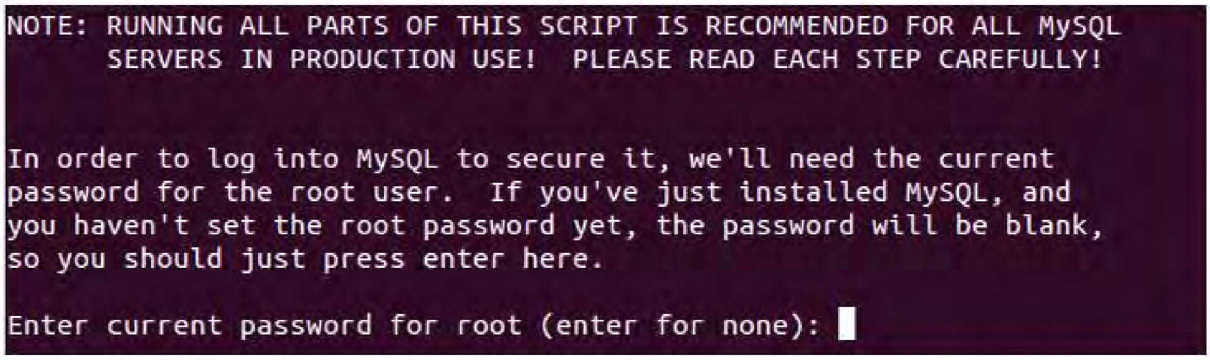
To do this run following command:

sudo mysql_secure_installation

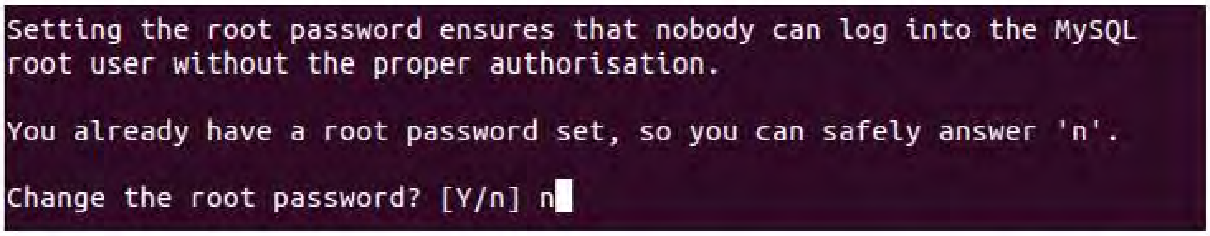


```
admin1@admin: ~
admin1@admin:~$ sudo mysql_secure_installation
```

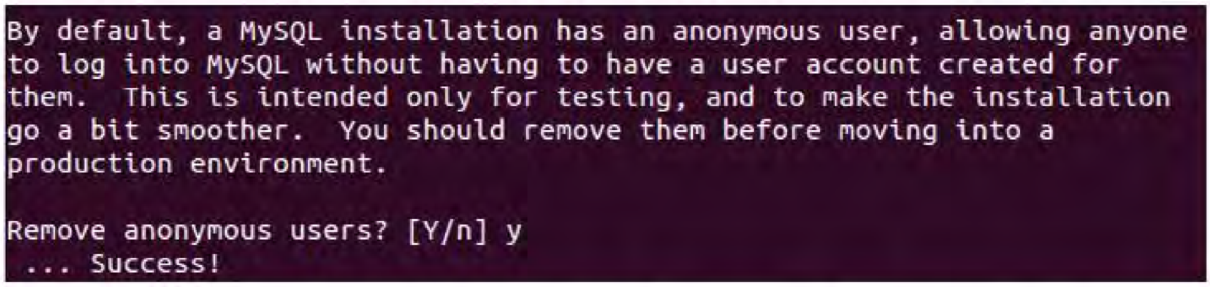
Now it will ask you for root password for your MySQL server. Complete the step by reading guidelines.



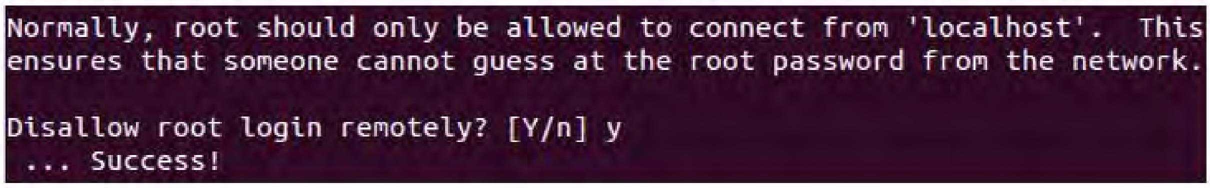
```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!


In order to log into MySQL to secure it, we'll need the current
password for the root user.  If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
```



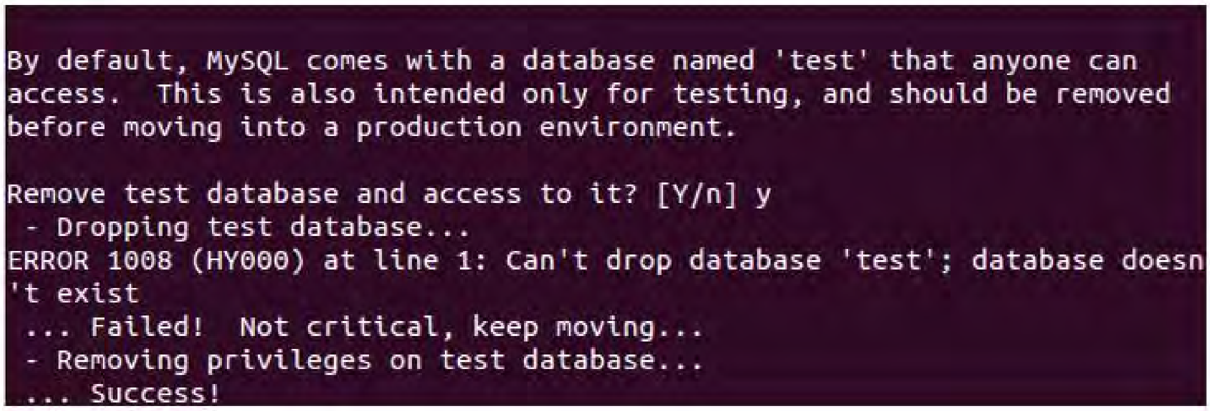
```
Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n
```



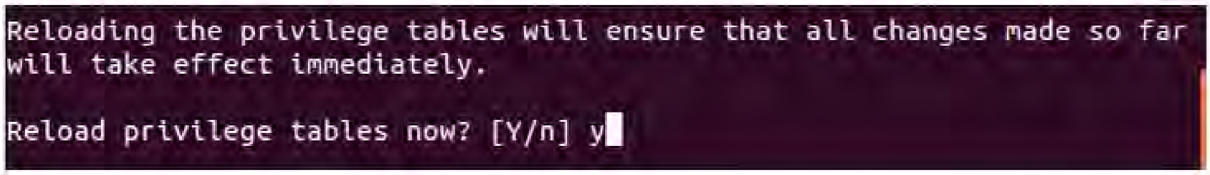
```
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!
```



```
Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!
```



```

By default, MySQL comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
ERROR 1008 (HY000) at line 1: Can't drop database 'test'; database doesn
't exist
 ... Failed!  Not critical, keep moving...
 - Removing privileges on test database...
 ... Success!
```


```
Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
```

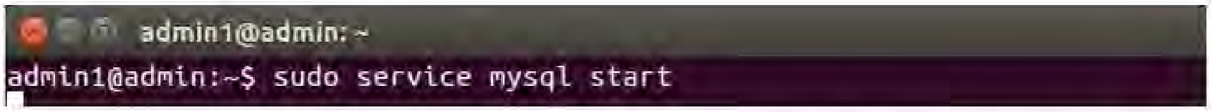
5.7 Starting and Testing MySQL

As we had installed our MySQL server now we want to start our server.

As we had seen previously we used service command to start apache web server. We will do the same for Mysql.

Run following command:

sudo service mysql start


```
admin1@admin: ~
admin1@admin:~$ sudo service mysql start
```

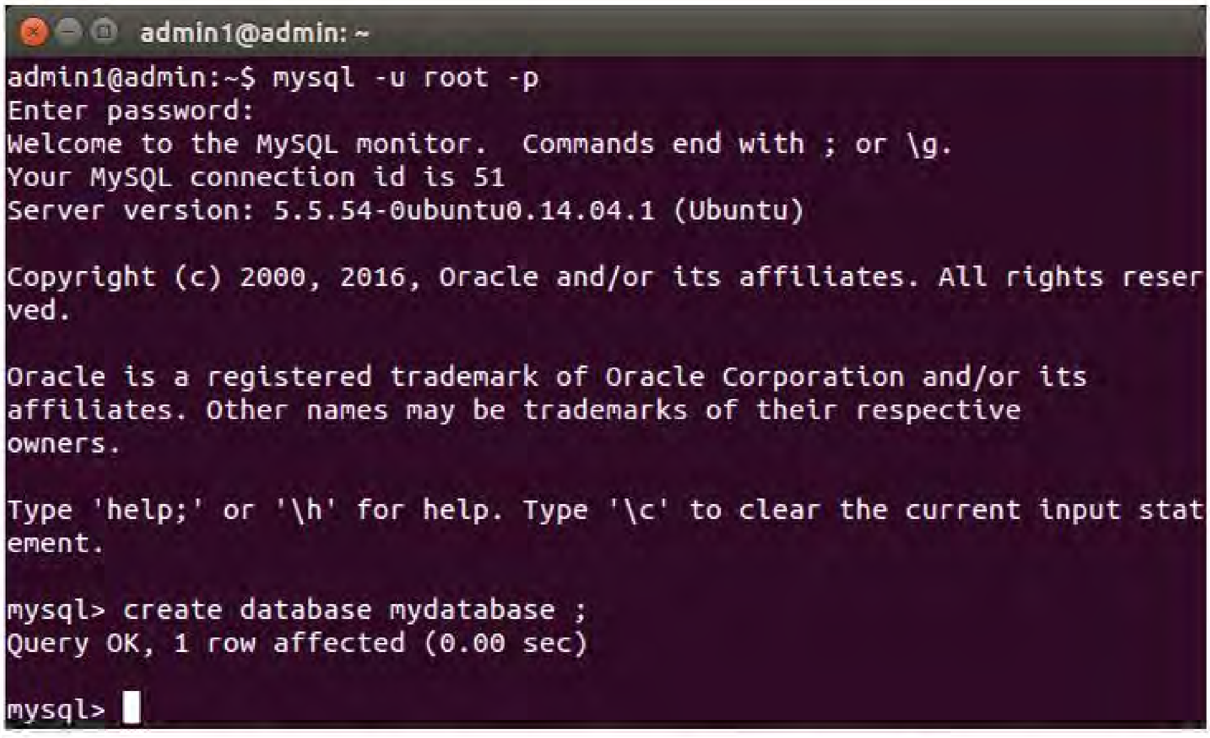
Now we will test our MySQL server. To do this we will use MySQL client.

First we will login to do this run following command:

Syntax:

mysql -u <user-name> -p <password>

mysql -u root -p



```
admin1@admin: ~
admin1@admin:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 51
Server version: 5.5.54-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reser
ved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
ement.

mysql> create database mydatabase ;
Query OK, 1 row affected (0.00 sec)

mysql> 
```

When prompted enter the root password you assigned mysql_secure_installation command script was run.

You will see "mysql>" prompt.

To list all command you can use. run following command:

mysq>\h

It will list all command available for use.

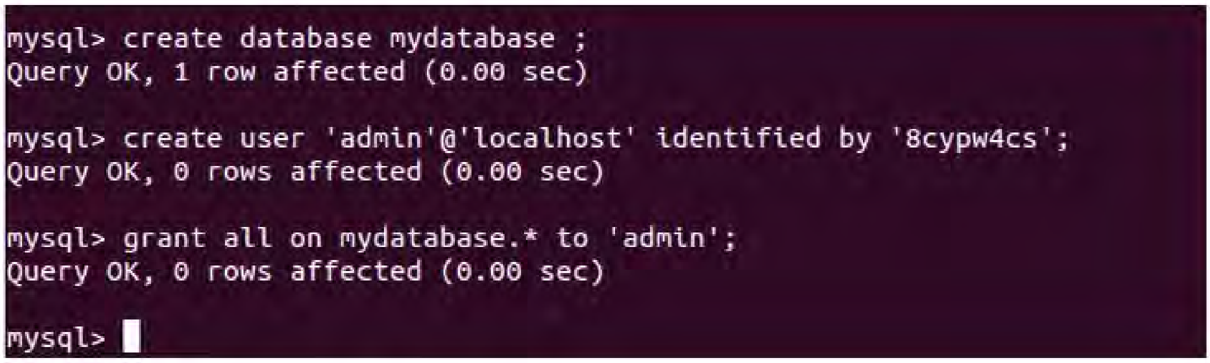
Now we will create our first MySQL user and First Database for testing.

To create new Database run following command:

create database mydatabase;

create user 'admin'@'localhost' identified by 'YourPasswordHere';

grant all on mydatabase.* to 'admin';


```
mysql> create database mydatabase ;
Query OK, 1 row affected (0.00 sec)

mysql> create user 'admin'@'localhost' identified by '8cypw4cs';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on mydatabase.* to 'admin';
Query OK, 0 rows affected (0.00 sec)

mysql> 
```

To exit run:

exit

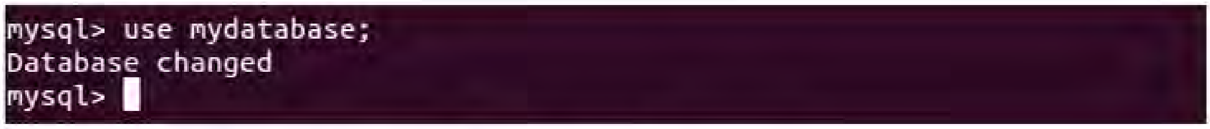
Now login as admin in mysql

mysql -u admin –p

Enter password when prompted

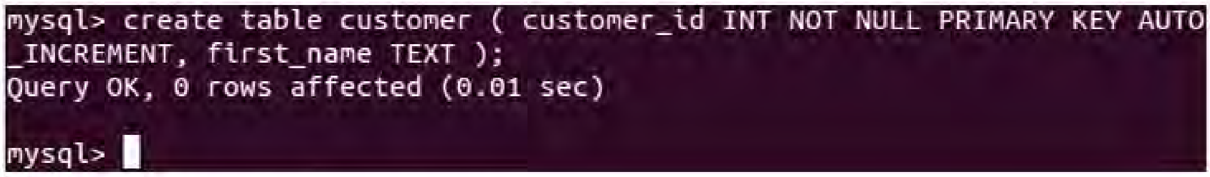
To select Database run following command:

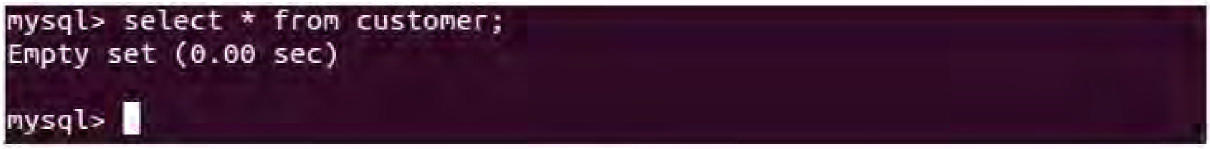
use mydatabase;


```
mysql> use mydatabase;
Database changed
mysql> 
```

To create table in 'mydatabase' run following command:

create table customer (customer_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, first_name TEXT);


```
mysql> create table customer ( customer_id INT NOT NULL PRIMARY KEY AUTO
_INCREMENT, first_name TEXT );
Query OK, 0 rows affected (0.01 sec)

mysql> 
```


```
mysql> select * from customer;
Empty set (0.00 sec)

mysql> 
```

To exit:

exit

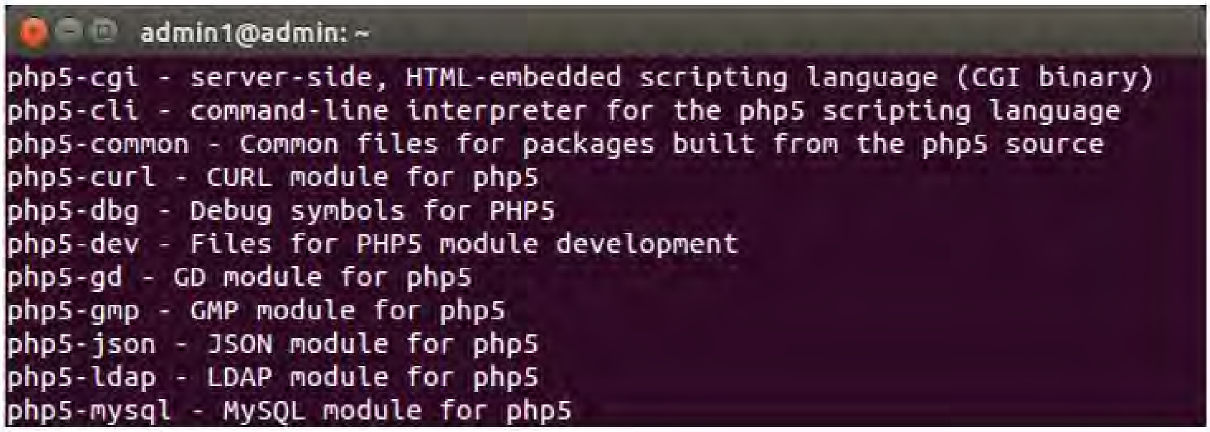
5.8 Installing the php-mysql Module

To add features and advance functionality we can install optional PHP modules and libraries.

To search all php modules we will use APT tool. But now command will be different.

To search php modules run following command:

sudo apt-cache search php5- | less



```
admin1@admin: ~
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
php5-cli - command-line interpreter for the php5 scripting language
php5-common - Common files for packages built from the php5 source
php5-curl - CURL module for php5
php5-dbg - Debug symbols for PHP5
php5-dev - Files for PHP5 module development
php5-gd - GD module for php5
php5-gmp - GMP module for php5
php5-json - JSON module for php5
php5-ldap - LDAP module for php5
php5-mysql - MySQL module for php5
```

It will list all the available PHP5 modules. To install modules use regular apt-get install command with module package name.

5.9 Checking the php-mysql Module

As we had installed php5 so we will search module for php5 version.

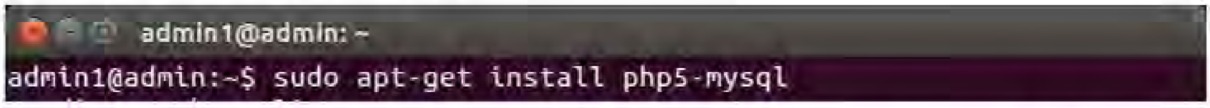
To search we will run following command:

sudo apt-cache search php5-mysql

It will list all the available module for php5.

To install this module we will run following command:

sudo apt-get install php5-mysql



```
admin1@admin: ~
admin1@admin:~$ sudo apt-get install php5-mysql
```

Now you have installed php5-mysql module, now you can connect to MySQL server from php files.

5.10 Self Learning Exercise

Q.1 What is Apache installing command?

a) sudo apt-get install apache.
b) sudo apt-get install apache2.
c) sudo apt-get install apacheHttpServer.
d) sudo apt-get install apacheWebServer.

Q.2 Why we use HostNameLookups?

a) Makes Apache perform 2 forward name lookups from different DNS servers to confirm the host name prior to logging.
b) Performs a host name lookup in the double file prior to logging the result.
c) Performs both a reverse name lookup and then a forward name lookup on that result, prior to logging the result.
d) Sets the hostname to double for all host names logged.

Q.3 Why we use **SUDO command**?

a) To get root access for particular command.
b) To get Super User privilege.
c) To switch between multiple user.
d) To change user password.

Q.4 How to check Apache version?

a) By starting Apache in Verbose mode and start logging.
b) We can’t know the Apache version.
c) By using –v flag.
d) By using sudo command.

Q.5 What does the mod_dir Module provide?

a) It provides a basic file directory searching capability for Apache to resolve spelling mismatches.
b) It provides trailing slash “/” redirects and serving directory index files.
c) It allows remote synchronization for tools such as Microsoft FrontPage.
d) It has been discontinued as of Apache 1.2 and is no longer used.

5.11 Answer to Self-Learning Exercise

1. b
2. c
3. a
4. c
5. b

5.12 Summary

In this unit we saw how to install Apache, PHP & MySQL in Linux environment. How we can configure the apache installation to start interpreting php code. We also learned how we can make our database and tables in it.

5.13 Glossary

Absolute pathname: any LINUX pathname that begins with a slash (/) indicating that path starts from the root.

1. Relative pathname: the pathname relative to the current directory. A relative pathname should not be stated with a slash (/).
2. Sudo: it is used to get administrative privilege for particular command.
3. Nautilus: it is the default file explorer for Ubuntu operating system, you have to run it with sudo command in order to make changes in system directory.

5.14 Exercise

Q.1. What PHP stands for earlier?

a) Hypertext Preprocessor
b) Pre Hypertext Processor
c) Personal Home Page
d) Pre Hypertext Process

Q.2. Which of the following tags is .... a valid way to begin and end a PHP code block.

Q.3. How does the identity operator === compare two values?

a) It converts them to a common compatible data type and then compares the resulting values
b) It returns True only if they are both of the same type and value
c) If the two values are strings, it performs a lexical comparison
d) It bases its comparison on the C strcmp function exclusively
e) It converts both values to strings and compares them

Q.4. Under what circumstance is it impossible to assign a default value to a parameter while declaring a function?

a) When the parameter is Boolean
b) When the function is being declared as a member of a class
c) When the parameter is being declared as passed by reference
d) When the function contains only one parameter

Q.5. Variables always start with a........ in PHP

a) Pond-sign
b) Yen-sign
c) Dollar-sign
d) Euro-sign

Q.6. MySQL runs on which operating systems?

a) Linux and Mac OS-X only
b) Any operating system at all
c) Unix, Linux, Windows and others
d) Unix and Linux only

Q.7. To remove duplicate rows from the result set of a SELECT use the following keyword:

a) NO DUPLICATE
b) UNIQUE
c) DISTINCT
d) None of the above

Q.8. Which of the following can add a row to a table?

a) Add
b) Insert
c) Update
d) Alter

Q.9. To use MySQL on your computer, you'll need?

a) FTP and Telnet
b) Some sort of client program to access the databases
c) A Browser
d) Perl, PHP or Java

Q.10. Which SQL statement is used to insert a new data in a database?

a) INSERT INTO
b) UPDATE
c) ADD
d) INSERT NEW

5.15 Answer to Exercise

1.	C	2.	<?php ?>
3.	A	4.	C
5.	C	6.	C
7.	C	8.	A
9.	B	10.	A

Reference and Suggested Reading

1. N.P.Gopalan, J. Akilandeswari, "Web Technology - A Developer's Perspective", PHI Learning Private Limited.
2. Kogent Learning Solutions Inc., "Web Technologies: HTML, JAVASCRIPT, PHP, JAVA, JSP, ASP.NET, XML and Ajax, Black Book: HTML, Javascript, PHP, Java, Jsp, XML and Ajax, Black Book", Dreamtech Press.
3. Ivan Bayross, "Web Enabled Commercial Application Development Using HTML, DHTML, PERL, Java Script", BPB Publications.

UNIT-6
PHP: Hypertext Pre-processor

Structure of the Unit

6.0 Objective

After reading this chapter you will be able to understand the commonly used server side scripting language, PHP and by using PHP you will able to do the following:

- Understand the basic concept of server side scripting language.
- Create the environment for developing Web pages using PHP.
- Understand the basic building block to code in PHP.
- Create the dynamic web pages with the help of PHP and MySQL.

6.1 Introduction

In the later chapters, we have learned several web based languages like HTML, JavaScript etc. Now we will move to the new Scripting language called PHP. PHP stands of "PHP: Hypertext Preprocessor". PHP is Server Side Scripting Language that means the scripting code for PHP is executed on server. It is also noted that PHP is interpreted language, means that to process by PHP engine. PHP is free to download and use. It is noted that before learning PHP you should familiar with HTML, CSS and JavaScript. PHP code is generally stored in a file with the file extension of ".php". You can use a simple text editor to write all the PHP code and saved with the file extension of ".php". In this chapter we used gedit as a text editor, which is the default editor in Ubuntu.
Before starting PHP programming, let's starts with the fundamentals.

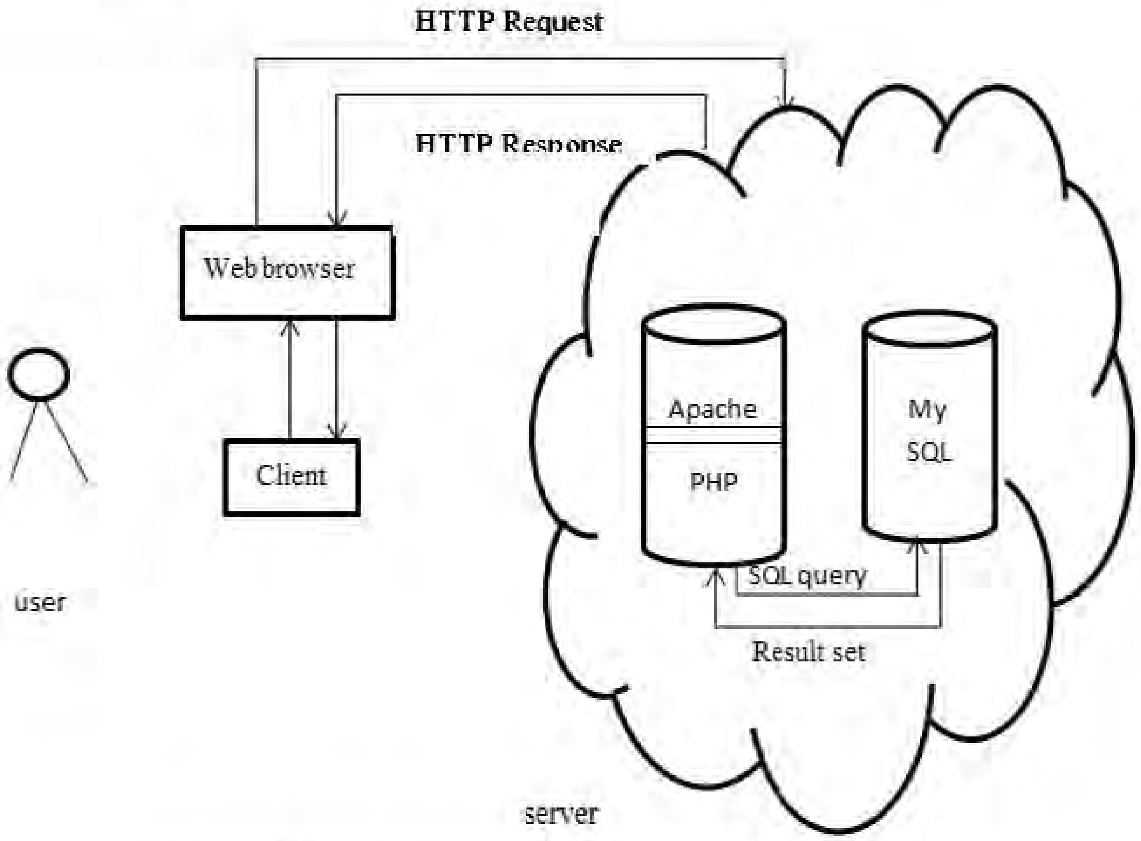


Figure 6.1: Fundamentals of server Scripting language

For understanding the fundamentals of server side scripting language, we take an example as:

Suppose use opens a web browser and type the URL to the website. Now browser sends an HTTP request to the corresponding server IP address, corresponding server will receive the request and the URL ends with .php extension will automatically redirect to the php layer , where it invoke the php interpreter.The php interpreter passes the files, .execute the code which are inside the php tag, interact with MySQL if there is requirement.
When the PHP interpreter complete the execution of php script then it returns the result to the web browser by HTTP response and the result is shown on the user's web browser.

6.2 Getting Started with PHP

PHP is simple, efficient, secure, flexible and free to use server side scripting language. It can be used to handle forms, managing data in databases, encrypting data, accessing cookies and many more.

The PHP code is enclosed within the start "<?php" and end with "?>" processing tag as

```
<?php
------Write PHP Code here -------
?>
```

Comments

The syntax of comments is like C, C++ comments. The comment in PHP code can be used as shown below:

```
<?php
// This is a single line comment in PHP
/*
This is Multi
Line comment


in PHP
*/
```

```
?>
```

Just like C and C++, in PHP every instruction is separated with the semicolon ";" at the end of each statement.

Ex:

```
<?php
echo "Hello World";
?>
```

Its syntax is C-Like.

6.3 Data Types and Variables

Variables:-

Variable is used to store various type of values. In PHP, variable is start with $ sign followed by the name of variable. It is case sensitive. You can assign a value using assignment operator "=". Unlike C, C++ and Java, you can assign any type of data into the variable in PHP. The PHP parser can convert that variable into the corresponding type. That's why PHP is also called loosely type language.

Example:

```
<?php
$var1 = 123;
$var2 = "VMOU KOTA";
$var3 = 123.456;
?>
```

In programming languages, data type is the classification of data of various types such as integer, character etc. PHP supports eight types of primitive data types which can be divided into the following categories:

i. Scalar data type – Boolean, integer, float (double), and string.

ii. Compound data type – array, and object.

iii. Special data type – resource, and NULL.

In PHP, the data type of variable is not set by the programmer, it is set when programmer assigned a value in a variable.

Boolean

This is the simplest type. In this type, the value of the variable is either TRUE or FALSE. It is generally used in logical statements.

Syntax: To specify a Boolean literal, use the keywords TRUE or FLASE. Both are case- insensitive.

Example:

```
<?php
//assigning a Boolean value into a variable.
$my_var1= TRUE;
$my_var2=False;
?>
```

Integer

An integer is whole number with the range {---------, -2, -1, 0, 1, 2, -----------}.

The integer can be specified in the following notations:

- Decimal Notation (base 10): represent using number.
- Octal Notation (base 8): represent by preceding the number with "0".
- Hexadecimal Notation (base 16): represent by preceding the number with "0x".
- Binary Notation (base 2): represent by preceding the number with "0b".

Example:

```
<?php
$var1 = 31; // positive decimal number
$var2 = -51; // negative decimal number
$var3 = 0234; // Octal number
$var4 = 0x 1BF; // Hexadecimal number
$var5 = 0b110110; // Binary number
```

```
?>
```

Float (Double)

Float (Double) are the real values like 2.334, 5.66 etc. By default, float display the minimum places need in decimal number.

Example:

```
<?php
$var1= 3.19999;
$var2 = 1.211111;
$var3=$var1+$var2;
echo $var3; // prints 4.4
?>
```

The variable $var3 contain 4.4

String

The string are the sequence of characters. When we insert the value in single quote or double quote to a variable then the variable is treated as string type.

Example:

```
<?php
$var1 = “Hello World”;
$var2 = ‘Hello Again’;
$var3 = “”;  //  string with zero character.
echo $var1;
echo “<br>”;
echo $var2;
?>
```

The length of the string has no limit within the bound of the available memory.

Array

Array variable contains multiple value of same type at the same time. Suppose we want to store 100 numbers then instead of using 100 variables, we can use array of 100 length.
There are three kinds of array and each array can be accessed using IDs:

Numeric array: These array can store number, string and any object but their index will be represented by numbers. By default index starts from zero.

Example: First method to create array:

```
<?php

$MNC=array("Infosys","TCS","Adobe","IBM");

for($i=0; $i<count($MNC);$i++) //count function counts the elements of array i.e
4.

{

echo $MNC[$i];

echo "<br>"

}

?>
```

Example: Second method to create array

```
<?php

$MNC[0] = "Infosys";

$MNC[1] = "TCS";

$MNC[2] = "Adobe";

$MNC[3] = "IBM";
```

```
?>
```

Associative array: In this type of array, the index of array will be represented by named key.

Example:

```
<?php

$age={“Neeraj”=>”27”, “Sandeep”=>”29”, “Sushil”=>”33”};

?>
```

Example:

```
<?php

$age[“Neeraj”]=27;

$age[“Sandeep”]=29;

$age[“Sushil”]=33;

foreach($age as $x=>$x_value) // using a loop for associative array

{

echo “Index value == ”. $x. “array value ==”.$x_value;

echo “<br>”;

}

?>
```

Multidimensional array:

Previously we studied a single dimensional array, which can store a list of values. There are the situations where a table of values will have to be stored. Consider the following table, which is an example of 2-D array.

Name	Roll Number	Mobile Number
X	1	9414036124
Y	2	9414036311
Z	3	9571088835

In PHP, the above table can be stored in 2-D array as

```
$student = array(

array (“X”, 1, 9414036124),

array (“Y”, 2, 9414036124),

array (“Z”, 1, 9571088835)

);
```

The value in the 2-D array can be retrieved as

$student[0][0], $student[0][1]......

6.4 Constant and Operators:

Constant:

Constant is an identifier (name), which value cannot be changed during the script. Constant are automatically global for entire script.

Syntax for creating constant:

define(name, value, case-insensitivity)

where name is the name of constant

value is the value of constant

case-insensitivity represent whether the constant name is case-insensitive or not. Default is false.

Example:

```
<?php
define("UNIVERSITY", "Vardhman Mahaveer Open University", TRUE);
echo university;
?>
```

The above script echoes Vardhman Mahaveer Open University.

Operators:

They are used to perform certain operations on the operands

Some of the operators used in PHP are:

Categories of Operator	Operator	Example	Description
Arithmetic Operators	+	$a + $b	Arithmetic addition of values in variable $a and $b.
	-	$a - $b	Arithmetic subtraction of values in variable $a and $b.
	*	$a*$b	Arithmetic multiplication of values in variable $a and $b.
	/	$a/$b	Arithmetic division of values in variable $a and $b.
	%	$a%$b	Arithmetic modulus of values in variable $a and $b.
	**	$a**$b	This is exponential operator, which computer $a to the

			power $b.
Assignment Operators	=	$a=$b	The value of $b is copied into variable $a
	+=	$a+=$b	Addition of $a and $b is copied into $a
	-=	$a-=$b	Substraction of $a and $b is copied into $a
	*=	$a*=$b	Multiplication of $a and $b is copied into $a
	/=	$a/=$b	Division of $a and $b is copied into $a
	%=	$a%=$b	Modulo of $a and $b is copied into $a
Comparison Operators	==	$a==$b	Returns TRUE if value of $a and $b are equal otherwise FALSE.
	===	$a===$b	Returns TRUE if $a is equal to $b, and they are of the same type otherwise FALSE.
	!=	$a!=$b	Returns TRUE if value of $a and $b are not equal otherwise FALSE.
	<>	$a<>$b	Returns TRUE if value of $a and $b are not equal otherwise FALSE.

	!==	$a!==$b	Returns TRUE if $x is not equal to $y, or they are not of the same type otherwise FALSE.
	>	$a>$b	Returns TRUE if value of $a is greater than $b otherwise FALSE.
	<	$a<$b	Returns TRUE if value of $a is less than $b otherwise FALSE.
	<=	$a<=$b	Returns TRUE if value of $a is less than or equal to $b otherwise FALSE.
	>=	$a>=$b	Returns TRUE if value of $a is greater than or equal to$b otherwise FALSE.
Increment/Decrement Operators	++$x	++$x	Pre-increment
	$x++	$x++	Post-increment
	--$x	--$x	Pre-decrement
	$x--	$x--	Post-decrement
Logical Operators	and	$x and $y	TRUE if both $x and $y are true
	or	$x or $y	TRUE if either $x or $y is true
	xor	$x xor $y	TRUE if either $x or $y is true, but not both

	&&	$x && $y	TRUE if both $x and $y are true
	||	$x and $y	TRUE if either $x or $y is true
	!	!$x	TRUE if $x is not true
String Operators	.	$x.$y	Concatenation Opration
	.=	$x .= $y	Appends $x to $y
Array Operators	+	$x + $y	Union of $x and $y
	==	$x ==$y	Returns true if $x and $y have the same key/value pairs
	===	$x==$y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
	!=	$x !=$y	Returns true if $x is not equal to $y
	<>	$x <> $y	Returns true if $x is not equal to $y
	!==	$x !==$y	Returns true if $x is not identical to $y

6.5 Control Structure and functions

Control Structure are those which changes the sequential flow of executing the scripts.

Conditional Statements:

The following are some conditional statements in PHP:

if Statement:

The if statement is a powerful tool to make a decision and used to control the flow of execution of the statements.

Syntax:

```
if (Condition)
{
…..
//PHP Statements;
…….
}
```

if the condition in if statement is TRUE then the statements inside the is statement is executed otherwise these are not executed.

Example:

```
if($age>18)
{
echo "You are 18+";
}
```

In the above example, if the value of $age is greater than 18 then the string "You are 18+" prints on the web page.

if….else Statement

In this type of statement, if the condition is TRUE then statement1 is executed else statement2 is executed.

Syntax:

```
if(condition)
{
…...
//statement 1;
…...
}
```

```
else
{
……
//statement 2;
…….
}
```

Example:

```
if($percentage >=36)
{
echo "You are PASS";
}
else
{
echo "You are FAIL";
}
```

In the above example, if the value of $percentage is greater than or equal to 36 then the string "You are PASS" is printed otherwise the string "You are FAIL" is printed on the web page.

if…..elseif….else Statement

This type of statement are used when there are multiple conditions are present.

Syntax:

```
if(condition 1)
{
….
//statement 1;
….
}
else if(condition 2)
{
….
//statement 2;
….
}
…..
…..
else
{
…..
//statement n;
…..
}
```

Example:

```
if($percentage >= 80)
{
echo "GRADE A";
}
else if ($percentage<80 && $percentage>=60)
{
echo "GRADE B";
}
else if ($percentage<60 && $percentage >=36)
{
echo "GRADE C";
}
```

```
else
{
echo “FAIL”;
}
```

switch Statement

When there are many conditions is to be selected then the complexity of if...elseif….else statement increases. To tackle the complexity, PHP has build-in multiple decision statement known as switch statement.

Syntax:

```
switch(n)
{
case label1:
statement 1;
break;
case label2:
statement 2;
break;
…..
…..
default:
statement n;
}
```

When the value of n is match with the respective label then the matched block is executed. Use of break statement prevents execution of the code of next block.

Looping Statements

The concept of loops is used when we need to repeat of statements of codes. It is important when we need repetitive task within a program. The following are the looping statements in PHP:

While Loop: This type of loop repeats continuously while a prescribed condition is TRUE.

Syntax:

```
while(condition)
{
//repetative code;
}
```

Example:

```
$count=1;
while ($count<10)
{
echo "Hello";
$count++;
}
```

The above code will echo hello until the condition $count<10 is TRUE.

For Loop: PHP support some sophisticated type of loop like for loop, which initiate the counter value and increment/decrement the counter within the loop statement.

Syntax:

```
for (init;condition;increment/decrement)
{
//repetitive code;
}
```

Example:

```
for ($count=0;$count<10;$count++)
{
echo $count;
}
```

The above code will echo the value of $count i.e 0 to 9.

6.6 Connecting to MySQL using PHP

You can connect to database and manipulate the data inside database using PHP. For this purpose, we use MySQL database system. MySQL database system is open source, free to use, web based, fast, reliable and easy to use.

The data in MySQL is stored in tables. Tables are the collection of interrelated data and consist of rows and columns.

Generally, there are two approaches of connecting MySQL using PHP:

- Object-oriented approach
- Procedural approach

For the sake of simplicity, we follow procedural approach.

In order to connect MySQL to PHP, first we need to open the connection. We can open the connection through PHP by using the following function:

```
mysqli_connect($servername, $username, $password, $dbname);
```

Here,

$servername represent the name of server.

$username represent the name of User.

$password represent password.

$dbname represents name of database.

This function returns FALSE if the connection is not established and if we want to see the error in connection to MySQL you can echoes mysqli_connect_error() function.

Example:

```
<?php
$servername=”localhost”;
$username=”username”;
$password=”password”;
$conn=mysqli_connect($username, $username, $password);
if($conn)
```

```
{
die("Connection Failed:".mysqli_connection_error());
}
else
{
echo "Connection Successful";
}
?>
```

Here, die function is used to print a message and exit from the current script.

After successful opening of connection, we can create, add, edit and delete data in table, database and table itself.

Some of the SQL queries are shown in table:

	SQL query syntax	Example	Description
For Creating Database	CREATE DATABASE databasename	CREATE DATABASE college	This will create a database named college
For Creating Table	CREATE TABLE tablename [(columnname datatype(size), columnname datatype(size), ---)]	CREATE TABLE student (sid int, sname varchar(30), fname varchar(30), mobile int, email varchar(30));	This will create a student table which contains sid, sname, fname, mobile, email as a column name.
For inserting data into table	INSERT INTO tablename (columnname, columnname, ...) VALUES (expression, expression, ….)	INSERT INTO student (sid, sname, fname, mobile, email) VALUES (1, "sandeep", "hooda",	This will insert the value in a row.

		9828568741,"ghooda@gmail.com");	
For Selecting data from Table	SELECT columnname, columnname, ….. FROM table-name WHERE search-condition	SELECT * FROM student WHERE sid=1;	This will display all the information present in a table student whose sid is 1.
For deleting data from table	DELETE FROM tablename WHERE search-condition	DELETE FROM student WHERE sid=2;	This will delete the record of student whose sid is 2
For updating data in Table	UPDATE table-name SET column-name=expression, column-name=expression …..	UPDATE student SET name = "sushil" where sid=1	This will update the name to sushil whose sid = 1.

For running the queries of SQL we can use the following function:

mysqli_query($conn, $sql);

Where $sql is a string represents the SQL query.

$conn is the return value of mysqli_connect() function.

6.7 Building a Web Page using PHP

For better understanding, we take an example of simple web application which can:

- Insert data through HTML form.
- fetch/view the data contain in the database.

Suppose we have a HTML form with text field User Name, Age and Mobile Number. Database named “college” is already create and it contain a table “student”. The structure of “student” table is as follows:

Field Name	Data Type	Other Information
sid	integer	primary key
uname	varchar(50)	--
age	interger	--

connection.php

```
<?php
$servername = “localhost”;
$username = “root”;
$password = “123”;
$dbname= “college”;
$conn=mysqli_connect($servername, $username, $password, $dbname);
if (!conn)
{
die(“Connection Failed: ”. mysqli_connection_error());
}
?>
```

index.php

```
<html>
<head>
<title>User Information Form</title>
</head>

<body>
<form action=”insert-info.php” method = “POST”>
Name: <input type=”text” name=”uname”><br>
```

```
Age: <input type=”text” name=”age”><br>
Mobile Number: <input type=”text” name=”mnumber”><br>
</form>

</body>
</html>
```

insert-info.php

```
<?php
include_once "connection.php";
$sql="INSERT INTO student(uname, age, mnumber) VALUES
('".$_POST['uname']."', ".$_POST['age'].", ".$_POST['mnumber'].")";

if(!mysqli_query($conn, $sql))
{
die("Error in executing query".mysqli_error());
}
header("Location: index.php");
?>
```

show-info.php

```
<html>
<head>
<title>Showing Information of Student</title>
</head>
<body>
<h1>Student Information</h1>
<table border="1">
<thead>
<th>S.No.</th>
```

```
<th>Name</th>
<th>Age</th>
<th>Mobile Number</th>
</thead>
<?php
include_once "connection.php";
$sql="SELECT * FROM student";
if(!mysqli_query($conn, $sql))
{
die("Error in executing query".mysqli_error());
}
$i=1;
$result=mysqli_query($conn, $sql);
while($row=mysqli_fetch_array($result))
{
?>
<tr>
<td><?php echo $i;?></td>
<td><?php echo $row["uname"];?></td>
<td><?php echo $row["age"];?></td>
<td><?php echo $row["mnumber"];?></td>
</tr>
<?php $i++;} ?>
</table>
<body/>
</html>
```

6.8 Self Learning Exercise

Q.1 Which of the following file extension(s) is (are) used for a PHP file?

(a) .html (b) .js

(c) .xml (d) .php

Q.2 PHP is

(a) a web page (b) a scripting language

(c) a css file (d) a markup language

Q.3 For PHP code, which of the following processing tags are used?

(a) "<script>…</script>" tags (b) "<?php …?>"tags

(c) "<noscript>…</noscript>" tags (d) "<xml/></xml/>"tags

Q.4 Which of the following is a correct syntax for comments in PHP?

(a)<!—this is a HTML comment--> (b)/*this is a XML comment*/

(c) //this is a JavaScript comment (d)#this is a PHP comment

6.9 Summary

In this chapter we have learned the basics of server side scripting language called PHP. The basics include operators, variables, control structures etc have learned in that chapter. We also learned the connection between MySQL and PHP. In the end, we use these basics to build a web pages using PHP.

6.10 Glossary

PHP- is simple, efficient, secure, flexible and free to use server side scripting language.

MySQL- is Database Management system which is SQL base database.

6.11 Answer to Self Learning Exercise

Q.1	d	Q.2	b
Q.3	b	Q.4	b

6.9 Exercise

Q.1 What is server side scripting language? Explain.

Q.2 What are the various operator present in PHP? Explain.

Q.3 Why PHP is also known as loosely typed language?

Q.4 What is array? How different types of array is declared in PHP? Explain with suitable example.

Q.5 How various SQL queries is executed using PHP? Explain

Reference and Suggested Reading

1. N.P.Gopalan, J. Akilandeswari, “Web Technology - A Developer's Perspective”, PHI Learning Private Limited.
2. Kogent Learning Solutions Inc., “Web Technologies: HTML, JAVASCRIPT, PHP, JAVA, JSP, ASP.NET, XML and Ajax, Black Book: HTML, Javascript, PHP, Java, Jsp, XML and Ajax, Black Book”, Dreamtech Press.
3. Ivan Bayross, “Web Enabled Commercial Application Development Using HTML, DHTML, PERL, Java Script”, BPB Publications.

UNIT-7
Web Application Creation

Structure of the Unit

7.0 Objective

In this chapter we shall focus upon the following topics

- Creating a Web Application
- Learn about the MVC Design Pattern, Architecture & Coding
- Learn about the MVC Framework
- Learn about PHP Application on our MVC

7.1 Introduction

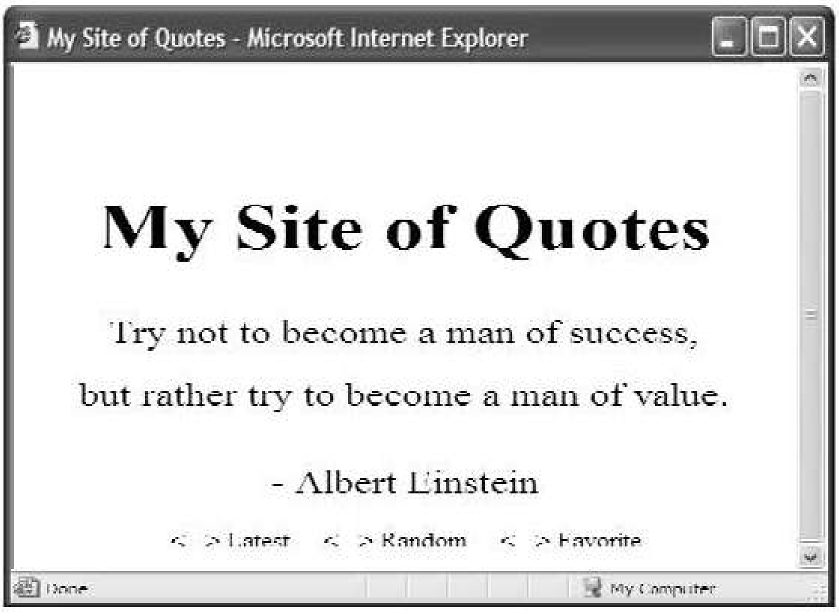
A web application is a program that runs on a computer with a web server, while its users interact with it via a web browser or similar user agent. Web application creation, also known as web development, is the creation of dynamic web applications. Examples of web applications are social networking sites like

Facebook or e-commerce sites like Amazon. A lot of people learn web coding because they want to create the next Facebook or find a job in the industry. But it's also a good choice if we just want a general introduction to coding, since it's super easy to get started. There are two broad divisions of web development – front-end development (also called client-side development) and back-end development (also called server-side development). Front-end development refers to constructing what a user sees when they load a web application – the content, design and how we interact with it. Back-end development controls what goes on behind the scenes of a web application.

7.2 Creating a Web Application – putting it all together

So, it seems that we need a group of Web site pages and tables, and an organized set of rules, that will give us the means to gather information, format it, and present it to the world for its enjoyment.

The first step when starting creating any web application is identifying the site's goals. The primary goal is to applying every thing we learned in earlier units. We will create a web application with combining the models from the previous units, creating a site that will store and display quotations. This site will use a MySQL database as the storage repository and the ability to create, edit, and delete quotations will be implemented.


My Site of Quotes - Microsoft Internet Explorer

My Site of Quotes

Try not to become a man of success,

but rather try to become a man of value.

- Albert Einstein

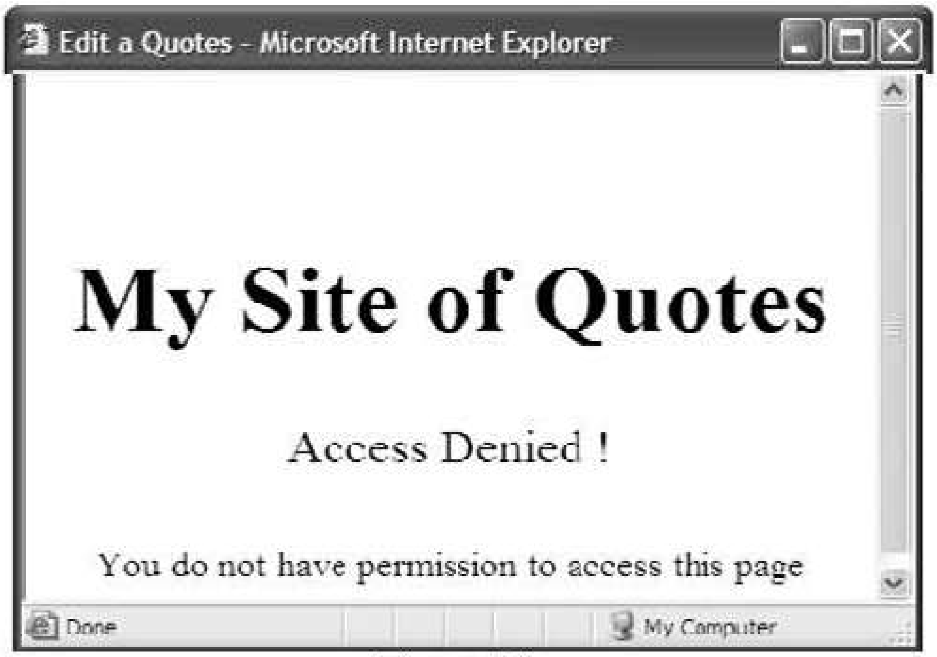
< > Latest < > Random < > Favorite

Done My Computer

Figure 7.1

Further, the public user will be able to view the most recent quotation by default in Figure 7.1 or a random one, or a random quotation previously marked as a favorite.

For improved security, the site will have an administrator who can log in and log out. And only the logged-in administrator will be allowed to create, edit, or delete quotations in Figure 7.2. The site will use a simple template to give every page a consistent look, with CSS handling all the formatting and layout. The site will also make use of one user-defined function, stored in an included file.



My Site of Quotes

Access Denied !

You do not have permission to access this page

Figure 7.2.

We should first create the database and its one table. The database could be named myquotes. We create its one table with this SQL command:

```
CREATE TABLE quotes (
quote_id INT UNSIGNED NOT NULL AUTO_INCREMENT,
quote TEXT NOT NULL,
source VARCHAR(100) NOT NULL,
favorite TINYINT(1) UNSIGNED NOT  NULL,
date_entered TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP, PRIMARY KEY (quote_id) )
```

The quote_id is the primary key, and it will automatically be incremented to the next logical value with each new submission. The quote field stores the quote itself, with the source field storing the attribution. The favorite field stores a 1 or a 0, marking the quote as a favorite or not. Finally, the date_entered column is a timestamp, automatically set to the current timestamp when a new record is created. We can create this table using a PHP script or a third-party application (like the MySQL client or phpMyAdmin).

Finally, a word about how the site should be organized on the server. Ideally the mysql_connect.php script, which establishes a database connection, would be stored outside the Web root directory. If that is not possible in our case, we can place it in the includes folder, then change the code in the other scripts accordingly.

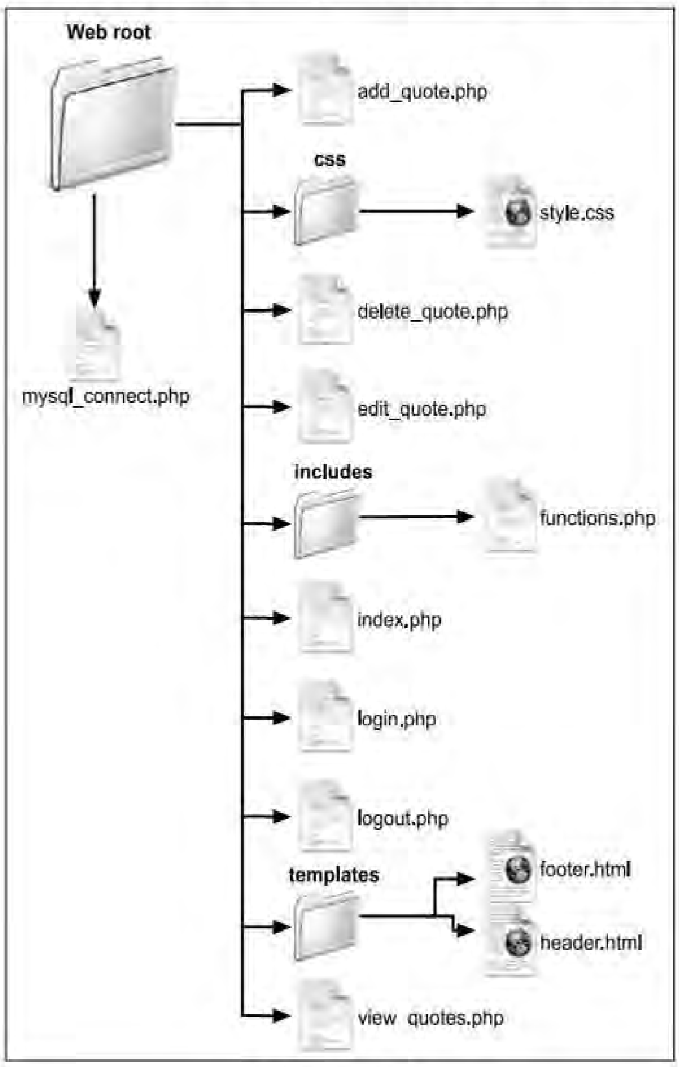


Figure 7.3

Connecting to the Database

Every script that interacts with the database which will be most but not all of them will then include this file (Figure 7.3). This file should be stored outside of the Web root directory or, if that's not possible, within the includes folder.

To create mysql_connect.php:

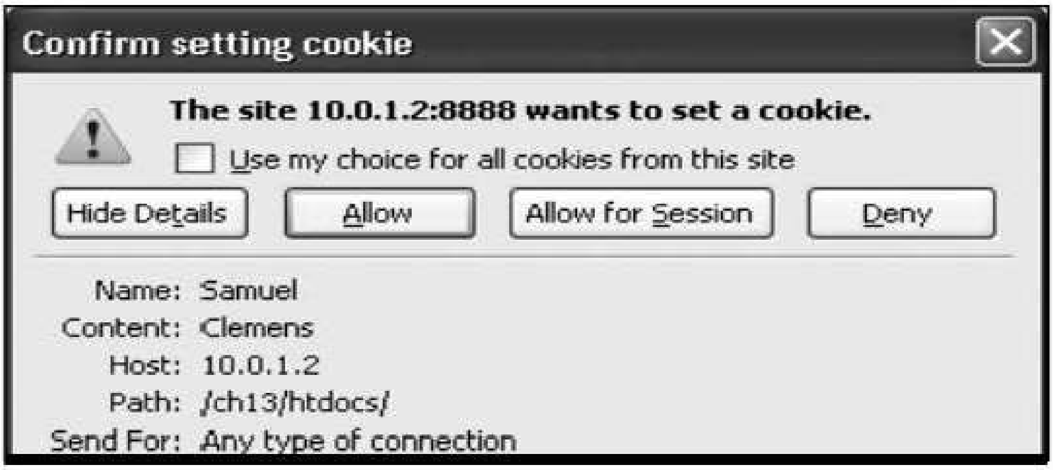
Begin a new PHP script in text editor or IDE, to be named mysql_connect.php:

Script 7.1

```
<?php // Script 7.1 - mysql_connect.php
/* This script connects to and selects the database. */

// Connect:
$dbc = mysql_connect('localhost', 'username', 'password');

// Select:
mysql_select_db('myquotes', $dbc);

?>
```

Writing the UserDefined Function

The site will have a single user-defined function. "Creating Functions," the best time to create our own functions is when a script or a site might have repeating code. In this site there are a couple such instances, but the most obvious one is this: Many scripts will need to check whether or not the current user is an administrator. In this next script, we'll create our own function that returns a Boolean value indicating if the user is an administrator. But what will be the test of administrative status? Upon successfully logging in, a cookie will be sent to the administrator's browser, with a name of Somesh and a value of *Clemens* (Figure 7.5). This may seem odd or random, and it is. When using something simple, like a cookie, for authentication, it's best to be obscure about what constitutes verification. If we went with something more obvious, such as a name of admin and a value of true, that'd be quite easy for anyone to guess and falsify. With this cookie in mind, this next function simply checks if a cookie exists with a name of Somesh and a value of *Clemens*.



The site 10.0.1.2:8888 wants to set a cookie.

☐ Use my choice for all cookies from this site

```
     Name:  Samuel
  Content:  Clemens
     Host:  10.0.1.2
     Path:  /ch13/htdocs/
 Send For:  Any type of connection
```

Figure 7.4

To create "functions" with named file functions.php (Script 7.2). The file must be save as functions.php and stored in the includes directory.

The function takes two arguments: the cookie's name and its value. Both have default values.

Since the function checks only a single cookie with a single expected value, it doesn't need to take arguments at all, let alone default ones. But by having the function take arguments, it can be used in different ways should the site's functionality expand (e.g., if we had multiple types of authentication to perform).

Script 7.2

```
<?php // Script 7.2 - functions.php
Begin defining a new function:

// This function checks if the user is an administrator.
// This function takes two optional values.
// This function returns Boolean value.
function is_administrator($name = 'Somesh', $value = 'Clemens') {

// Check for the cookies and check its value:
```

```
if (isset($_COOKIE[$name]) && ($_COOKIE[$name] == $value)) {
return true;
} else {
return false;
}

} // End of is_administrator() function.

?>
```

Creating the Template

Now that the two helper files have been created, it's time to move on to the template. While "Creating Web Applications," the site's layout will be controlled by two includable files: a header and a footer. Both will be stored in the templates directory. The header also references a style sheet, to be stored in the css folder.

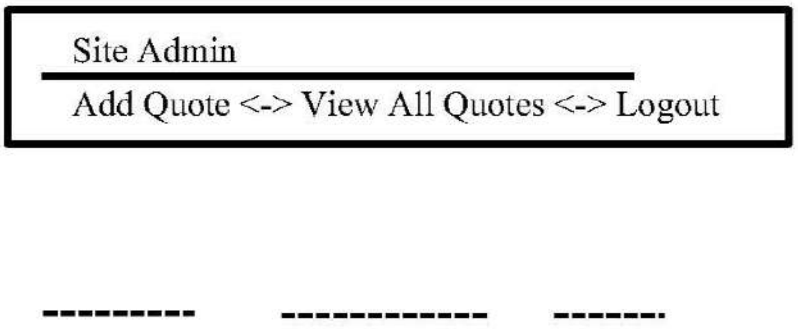


Figure 7.5

To create header.html:

Begin a new HTML document named with header.html (Script 7.3) and save in templates directory.

Script 7.3

```
<?php // Script 7.3 - header.html

```

```
// Include the functions script:
 include('includes/functions.php'); ?>
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" media="all" href="css/style.css" />
<title><?php // Print the page title.
if (defined('TITLE')) { // Is the title defined?
print TITLE;
} else { // The title is not defined.
print 'My Site of Quotes';
}
?></title>
</head>
<body>
<div id="container">
<h1>My Site of Quotes</h1>
<br />
<!-- BEGIN CHANGEABLE CONTENT. -->
```

To create footer.html:

Begin a new HTML document named with footer.html (Script 7.4) and save in templates directory.

This conditional is a bit complicated. To start, most pages can confirm that the current user is an administrator by just invoking the **is_administrator()** function. But because of how cookies work, that function will return inappropriate results on two pages: **login.php** and **logout.php**. On the **logout.php**, the script will have received the administrative cookie prior to deleting it. So on that page, it will seem like the user is an administrator (because they just were), but links to the administrative pages should not be shown as the user will be blocked from using them (because when they get to those pages, the cookie will no longer exist). Hence, the first part of the conditional requires that **is_administrator()** return true and that the current page not be **logout.php**. The code **basename($_SERVER['PHP_SELF'])** is a reliable way to get the current script (and because the footer file is included by another script, **$_SERVER['PHP_SELF']** will have the value of the including script. The second part of the conditional, after the OR, checks if the **$loggedin** variable is set and has a true value. This will be the case on the **login.php** page, after the user successfully logged in. The **is_administrator()** function won't return a true value at that juncture, because the cookie will have been just sent by the script, and therefore won't be available to be read.

Script 7.4

```
<!-- END CHANGEABLE CONTENT. -->
<?php // Script 7.4 - footer.html

 // Display general admin links...
// - if the user is an administrator and it's not the logout.php page
// - or if the $loggedin variable is true (i.e., the user just logged in)
if ( (is_administrator() && (basename($_SERVER['PHP_SELF']) != 'logout.php'))
OR (isset($loggedin) && $loggedin) ) {

 // Create the links:
print '<hr /><h3>Site Admin</h3><p><a href="add_quote.php">Add Quote</a> <->
 <a href="view_quotes.php">View All Quotes</a> <->
 <a href="logout.php">Logout</a></p>';

}

?>
</div><!-- container -->
<div id="footer">Content &copy; 2017</div>
</body>
</html>
```

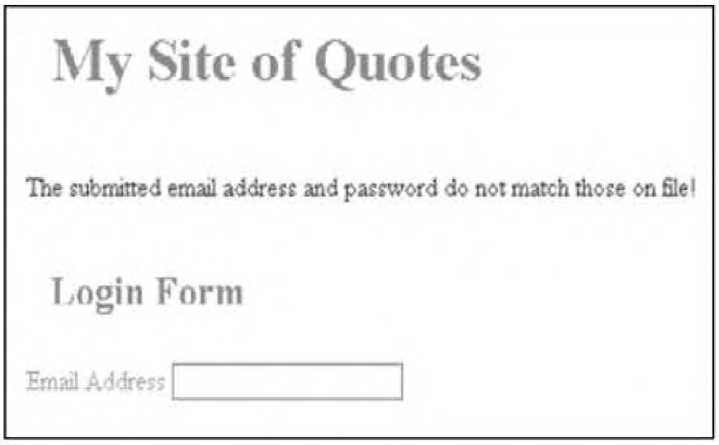
Logging In

Next, it's time to create the script through which the administrator can log in. This site does not use output buffering, so the handling of the form must be written in such a way that the cookie can be sent without generating headers already sent errors. Begin a PHP document named with login.php (Script 7.5):

Script 7.5

```
<?php // Script 13.5 - login.php
/* This page lets people log into the site. */

// Set two variables with default values:
$loggedin = false;
$error = false;

// Check if the form has been submitted:
if ($_SERVER['REQUEST_METHOD'] == 'POST') {

// Handle the form:
if (!empty($_POST['email']) && !empty($_POST['password'])) {

 if ( (strtolower($_POST['email']) == 'me@example.com') && ($_POST
['password'] == 'testpass') ) { // Correct!

// Create the cookie:
setcookie(‘Somesh', 'Clemens', time()+3600);

// Indicate they are logged in:
$loggedin = true;

} else { // Incorrect!

 $error = 'The submitted email address and password do not
match those on file!';

}

 } else { // Forgot a field.

 $error = 'Please make sure you enter both an email address and a
password!';

 }

 }

 // Set the page title and include the header file:
 define('TITLE', 'Login');
 include('templates/header.html');
```

```

// Print an error if one exists:
if ($error) {
print '<p class="error">' . $error . '</p>';
}

// Indicate the user is logged in, or show the form:
if ($loggedin) {

print '<p>You are now logged in!</p>';

} else {

print '<h2>Login Form</h2>
<form action="login.php” method="post">
<p><label>Email Address <input type="text" name="email"
/></label></p>
<p><label>Password <input type="password" name="password" />
</label></p>
<p><input type="submit" name="submit“ value="Log In!" /></p>
</form>';

}

include('templates/footer.html');  // Need the footer.
?>
```



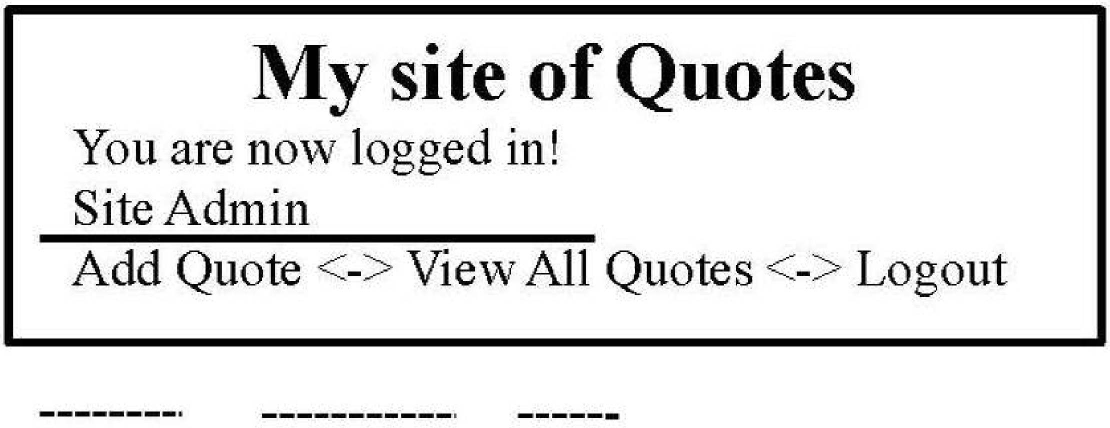
My Site of Quotes

The submitted email address and password do not match those on file!

Login Form

Figure 7.5

The error itself will be determined in Step 7, but it can't be printed at that point because the HTML header will not have been included. The solution is to have this code check for a non-false $error value and then print $error within some HTML and CSS (Figure 7.5.).

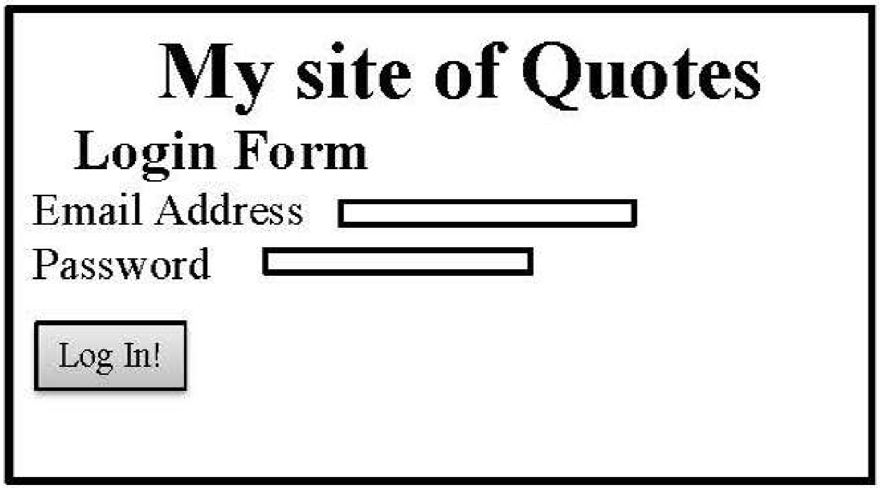


My site of Quotes

You are now logged in!

Site Admin

Figure 7.6



My site of Quotes

Login Form

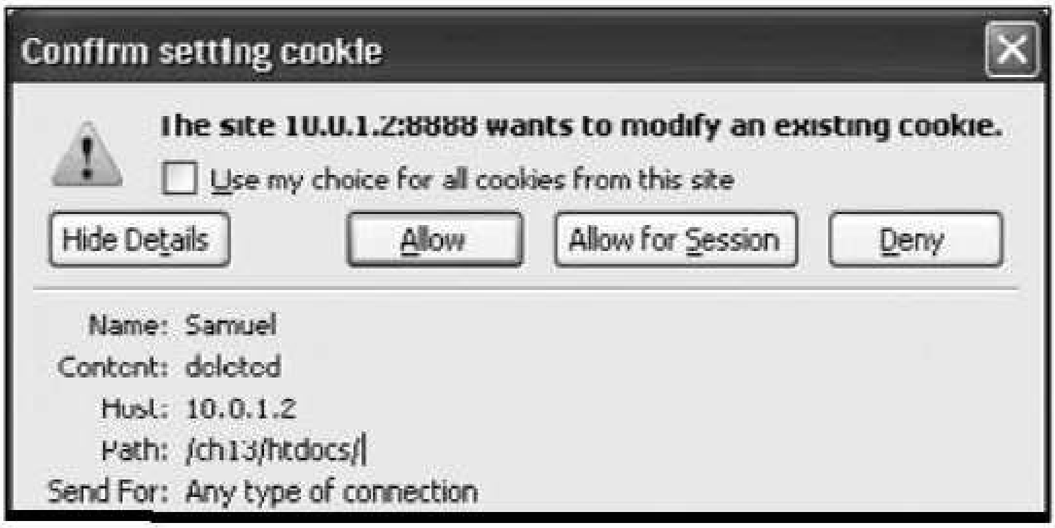
Figure 7.7

Logging Out

Begin a new PHP document named with **logout.php** (Script 7.6). Save and test the file in Web browser (Figure 7.8 and Figure 7.9)

Script 7.6

```
 <?php // Script 7.6 - logout.php
 /* This is the logout page. It destroys the cookie. */

 // Destroy the cookie, but only if it already exists:
 if (isset($_COOKIE['Somesh'])) {
setcookie('Samuel', FALSE,  time()-300);
 }

// Define a page title and include the header:
 define('TITLE', 'Logout');
 include('templates/header.html');

 // Print a message:
 print '<p>You are now logged out.</p>';

// Include the footer:
include('templates/footer.html');
?>
```

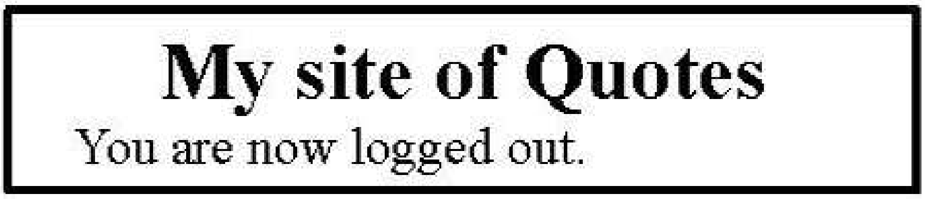

Confirm setting cookie

The site 10.0.1.2:8888 wants to modify an existing cookie.

Use my choice for all cookies from this site

Name: Samuel
Content: deleted
Host: 10.0.1.2
Path: /ch13/htdocs/
Send For: Any type of connection

Figure 7.8

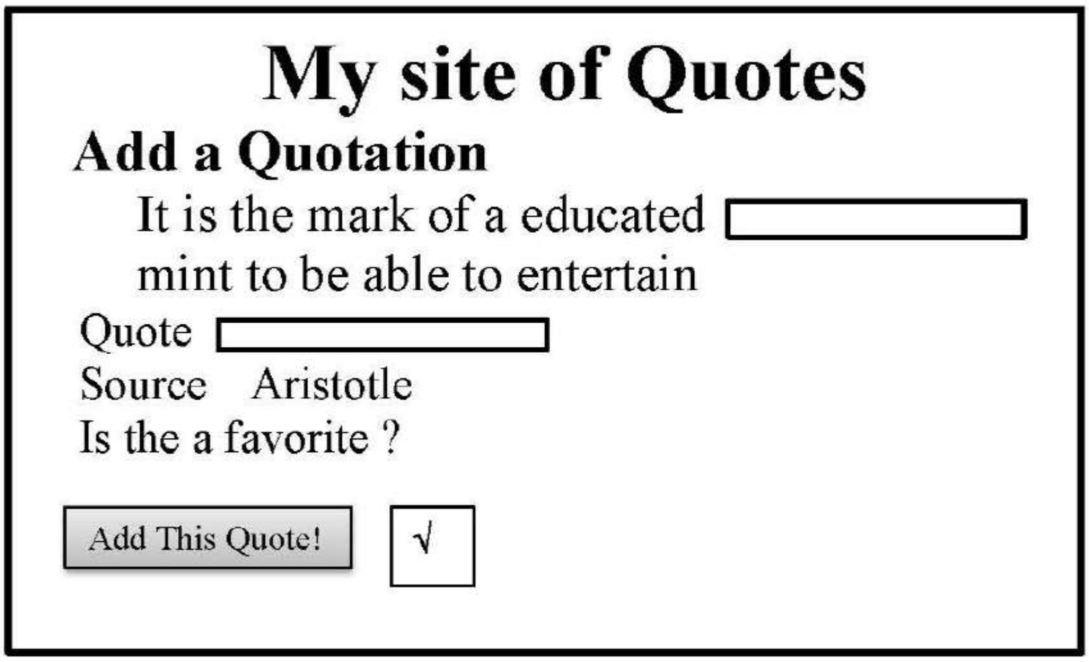

My site of Quotes

You are now logged out.

Figure 7.9

Adding Quotes

Now that the administrator has the ability to log in, she or he should be able to start adding quotations. The script for doing so is a lot like the one for adding blog postings, but the form will have an additional checkbox for marking the quotation as a favorite (Figure 7.10). Because this script creates records in the database, security must be a primary concern. As an initial precaution, the script will make sure that only the administrator can use the page. Second, values to be used in the SQL command will be sanctified to prevent invalid queries. Begin a new PHP document named with add_quote.php (Script 7.7) and test file.


My site of Quotes

Add a Quotation

It is the mark of a educated mint to be able to entertain

Quote

Source Aristotle

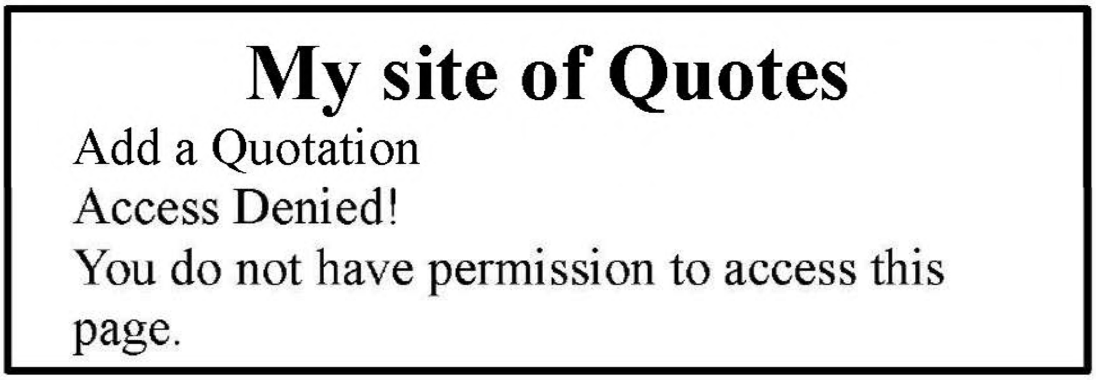
Is the a favorite ?

Figure 7.10

Script 7.7

```
<?php // Script 7.7 - add_quote.php
/* This script adds a quote. */

// Define a page title and include the header:
define('TITLE', 'Add a Quote');
include('templates/header.html');

 print '<h2>Add a Quotation</h2>';

// Restrict access to administrators only:
 if (!is_administrator()) {
 print '<h2>Access Denied!</h2><p class="error">You do not have permission    to access this page.</p>';
 include('templates/footer.html');
 exit();
 }

// Check for a form submission:
 if ($_SERVER['REQUEST_METHOD'] == 'POST') { // Handle the form.

 if ( !empty($_POST['quote']) && !empty($_POST['source']) ) {

// Need the database connection:
include('../mysql_connect.php');

// Prepare the values for storing:
 $quote = mysql_real_escape_string(trim(strip_tags($_POST['quote'])), $dbc);
$source = mysql_real_escape_string(trim(strip_tags($_POST['source'])), $dbc);

 // Create the "favorite" value:
 if (isset($_POST['favorite'])) {
$favorite = 1;
 } else {
 $favorite = 0;
 }

$query = "INSERT INTO quotes(quote, source, favorite) VALUES ('$quote',        '$source', $favorite)";
 $r = mysql_query($query, $dbc);

if (mysql_affected_rows($dbc) ==1) {
 // Print a message:
 print '<p>Your quotation has been stored.</p>';
} else {
print '<p class="error">Could not store the quote because:<br />' .
mysql_error($dbc) . '.</p><p>The query being run was: ‘. $query . '</p>';
}

 // Close the connection:
 mysql_close($dbc);

 } else { // Failed to enter a quotation.
 print '<p class="error">Please enter a quotation and a source!</p>';
 }

 } // End of submitted IF.

 // Leave PHP and display the form:
 ?>

 <form action="add_quote.php“ method="post">
<p><label>Quote <textarea name="quote" rows="5" cols="30“>
</textarea>    </label></p>
<p><label>Source <input type="text“ name="source" /></label></p>
<p><label>Is this a favorite? <input type="checkbox" name="favorite“
value="yes" /></label></p>
<p><input type="submit" name="submit“ value="Add This Quote!" /></p>
 </form>

```

```
<?php include('templates/header.html');
?     >
```



My site of Quotes

Add a Quotation

Access Denied!

You do not have permission to access this page.

Figure 7.11

Creating the Home Page

Home Page Last, but certainly not least, there's the home page. For this site, the home page will be the only page used by the public at large. The home page will show a single quotation, but the specific quotation can be one of the following:

- **The most recent (the default)**
- **A random quotation**
- **A random favorite quotation**

To achieve this effect, links will pass different values in the URL back to this same page. The script should also display administrative links—edit and delete—for the currently displayed quote, if the user is an administrator. Begin a PHP document named index.php (Script 7.8) and save the file and test in our web browser:

Script 7.8

```
 <?php // Script 7.8 - index.php
/* This is the home page for this site. It displays:
- The most recent quote (default)
- OR, a random quote
- OR, a random favorite quote */

// Include the header:
```

```
include('templates/header.html');

// Need the database connection:
include('../mysql_connect.php');

// Define the query...
// Change the particulars depending upon values passed in the URL:
 if (isset($_GET['random'])) {
 $query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY   RAND() DESC LIMIT 1';
 } elseif (isset($_GET['favorite'])) {
 $query = 'SELECT quote_id, quote, source, favorite FROM quotes WHERE         favorite=1 ORDER BY RAND() DESC LIMIT 1';
} else {
$query = 'SELECT quote_id, quote, source, favorite FROM quotes ORDER BY date_entered DESC LIMIT 1';
}

// Run the query:
if ($r = mysql_query($query, $dbc)) {

// Retrieve the returned record:
$row = mysql_fetch_array($r);

            // Print the record:
      print "<div><blockquote>{$row['quote']}</blockquote>-{$row['source']}";

```

```
    // Is this a favorite?
        if ($row['favorite'] == 1) {
        print ' <strong>Favorite!</strong>';
         }

        // Complete the DIV:
print '</div>';

// If the admin is logged in, display admin links for this record:
if (is_administrator()) {
print "<p><b>Quote Admin:</b> <a href=\"edit_quote.php?id
={$row['quote_id']}\">Edit</a> <->
<a href=\"delete_quote.php?id={$row['quote_id']}\">Delete</a>
</p>\n";
}

} else { // Query didn't run.
print '<p class="error">Could not retrieve the data because:<br />' .
mysql_error($dbc) .
'.</p><p>The query being run was: ' . $query . '</p>';
} // End of query IF.

mysql_close($dbc); // Close the connection.

print '<p><a href="index.php">Latest</a> <-> <a href="index.php?random
=true">Random</a> <-> <a
href="index.php?favorite=true">Favorite</a><p>';

```

```
include('templates/footer.html'); // Include the footer.
?>
```

7.3 The MVC Design Pattern – Basic Web Architecture

The MVC Design Pattern

Model-view-controller (MVC) is a pattern used to isolate business logic from the user interface. Using MVC, the Model represents the information (the data) of the application and the business rules used to manipulate the data, the View corresponds to elements of the user interface such as text, checkbox items, and so forth, and the Controller manages details involving the communication between the model and view. The controller handles user actions such as keystrokes and mouse movements and pipes them into the model or view as required.

MVC design pattern aims to separate content from presentation and data-processing from content. Theoretically sound, but where do we see this in MVC? One place is reasonably clear - between the data-processing (Model) and the rest of the application. The programming language Smalltalk first defined the MVC concept it in the 1970's. Since that time, the MVC design idiom has become commonplace, especially in object-oriented systems.

Basic Web Architecture

The web is a two-tiered architecture. A web browser displays information content, and a web server that transfers information to the client.

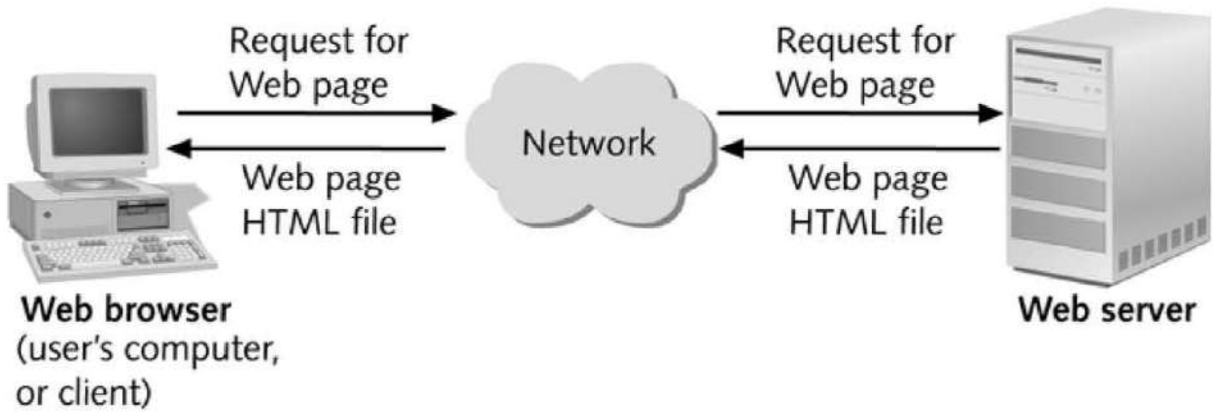


Figure 7.11

In the basic web architecture, as depicted in the figure above:

Web Browser-The primary purpose is to bring information resources to the user. An application for retrieving, presenting, and traversing information resources.
Web Server-The term web server or web server can mean one of two things:

- A computer program that accepts HTTP requests and returns HTTP responses with optional data content.
- A computer that runs a computer program as described above.

Though this architecture works perfectly well, it becomes difficult for maintaining the code as the web pages grow. With the growing web applications on the Web it became imperative for web application developers to seek refuge in application development frameworks that had the functional abstraction for ease of code readability and maintainability.

7.4 MVC Architecture, Coding Considerations

MVC Architecture

The Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. The MVC architecture pattern is made up of the following three parts:
The model is the central component of the pattern. It expresses the application's behavior. It is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.

A **View** can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants. They are script based templating systems like JSP, ASP, PHP and very easy to integrate with AJAX technology.
The **Controllers** act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. The controller receives the input, it validates the input and then performs the business operation that modifies the state of the data model.

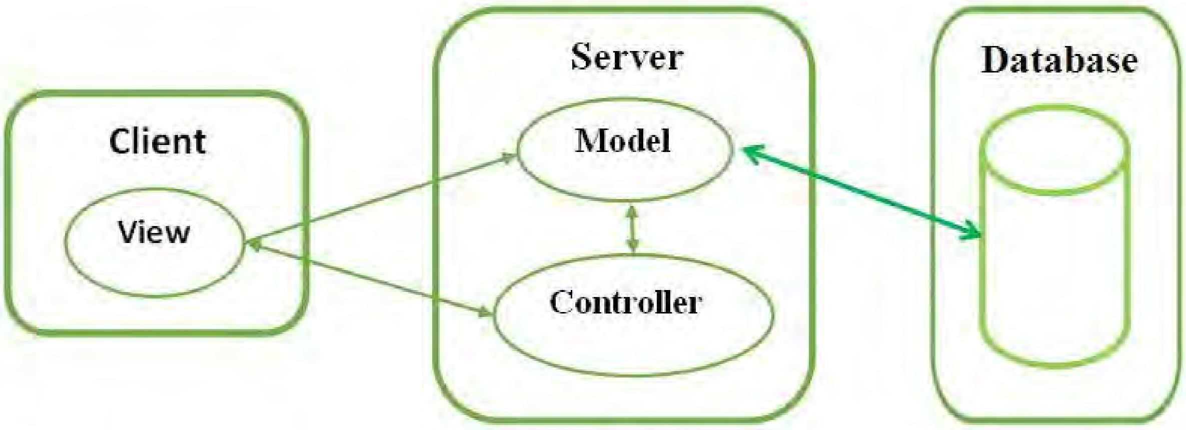


Figure 7.12

Coding Considerations

While writing code for our programs, we should consider the following points:

Name variables properly: Use short enough and long enough variable names in each scope of code. We should always avoid meaninglessly long names, which can defeat the purpose in the first place. The same would also apply while naming functions, methods, classes, etc. too.

Comments should be used carefully: Though comments are an important part in the whole "code understandability / readability" thing, we should never overdo it. Think about the developers who will be reading code. Comments should be used to complete the presentation of the code.I t should not be a daunting task for them.

Code Style and consistency: The style in which we prefer to write code should be consistent though out our applications. This makes the code both understandable and readable by the other developers who may be looking at our code.
Adhering to the above listed coding considerations not mandatory for writing code. However, this is the era of collaborative application development and making code readable and style consistent is the need of the hour.

7.5 Setting up our Development Environment

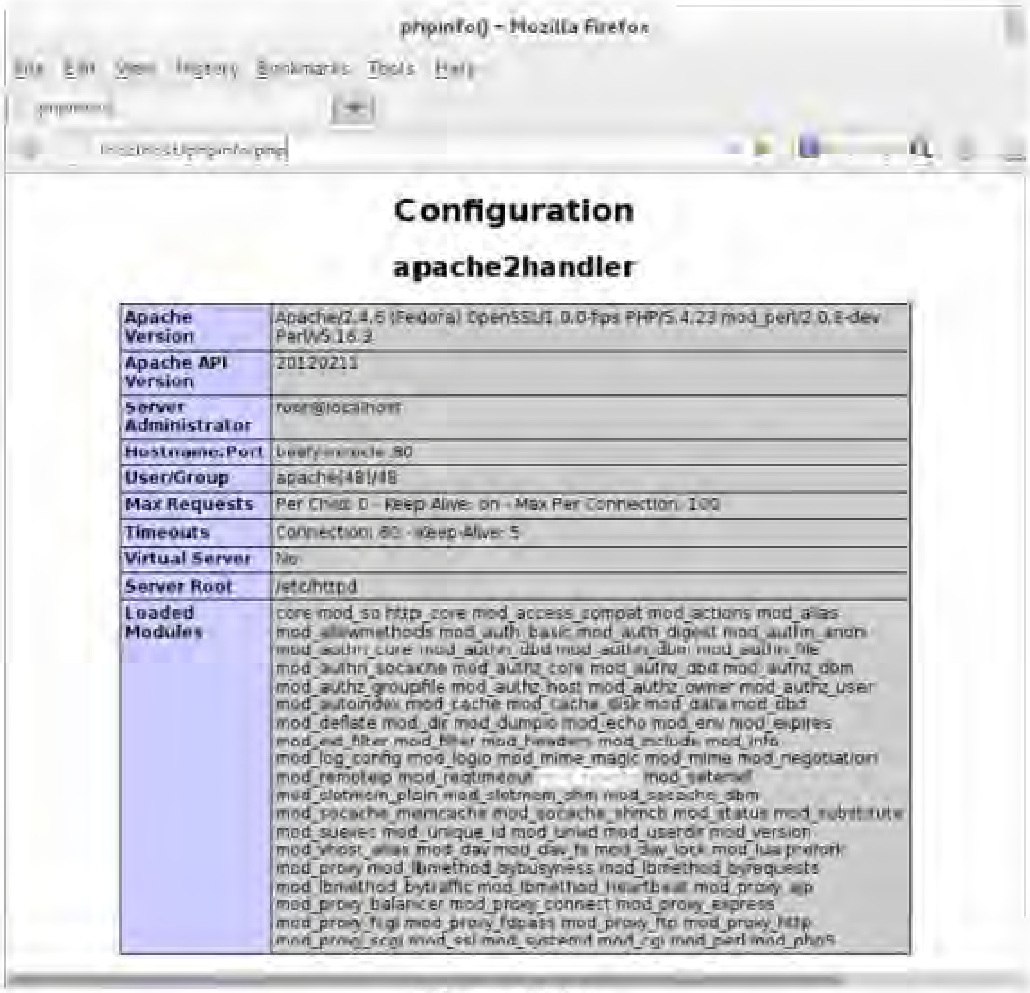
It's time to set up our development environment. This process is one of the most overlooked and oftentimes frustrating parts of learning to program, but it's easier with the right preparation.

A development environment is a set of software that enables we to write programs for a particular language or platform. This software oftentimes includes a text editor, shell, and a compiler/interpreter.

The following should be installed and available for use on Fedora 18 to proceed:

- Apache Web Server v2.4.x
- PHP/5.4.X
- MySQL 5.5.X

Some, additional steps that are needed to be performed are listed as below. The steps listed below are for the Fedora 18 distribution of Linux.


phpinfo() - Mozilla Firefox
File Edit View History Bookmarks Tools Help
phpinfo()
localhost/phpinfo.php

Configuration

apache2handler

Apache Version	Apache/2.4.6 (Fedora) OpenSSL/1.0.0-fips PHP/5.4.23 mod_perl/2.0.8-dev Perl/v5.16.3
Apache API Version	20120211
Server Administrator	root@localhost
Hostname:Port	beefy-miracle:80
User/Group	apache(48)/48
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 60 - Keep-Alive: 5
Virtual Server	No
Server Root	/etc/httpd
Loaded Modules	core mod_so http_core mod_access_compat mod_actions mod_alias mod_allowmethods mod_auth_basic mod_auth_digest mod_authn_anon mod_authn_core mod_authn_dbd mod_authn_dbm mod_authn_file mod_authn_socache mod_authz_core mod_authz_dbd mod_authz_dbm mod_authz_groupfile mod_authz_host mod_authz_owner mod_authz_user mod_autoindex mod_cache mod_cache_disk mod_data mod_dbd mod_deflate mod_dir mod_dumpio mod_echo mod_env mod_expires mod_ext_filter mod_filter mod_headers mod_include mod_info mod_log_config mod_logio mod_mime_magic mod_mime mod_negotiation mod_remoteip mod_reqtimeout [illegible] mod_setenvif mod_slotmem_plain mod_slotmem_shm mod_socache_dbm mod_socache_memcache mod_socache_shmcb mod_status mod_substitute mod_suexec mod_unique_id mod_unixd mod_userdir mod_version mod_vhost_alias mod_dav mod_dav_fs mod_dav_lock mod_lua prefork mod_proxy mod_lbmethod_bybusyness mod_lbmethod_byrequests mod_lbmethod_bytraffic mod_lbmethod_heartbeat mod_proxy_ajp mod_proxy_balancer mod_proxy_connect mod_proxy_express mod_proxy_fcgi mod_proxy_fdpass mod_proxy_ftp mod_proxy_http mod_proxy_scgi mod_ssl mod_systemd mod_cgi mod_perl mod_php5

Figure 7.13

- Confirm that the Apache mod_rewrite module is loaded and available. The ***/etc/httpd/conf.modules.d/00-base.conf*** file should contain the following

line:
LoadModule rewrite_module modules/mod_rewrite.so

The phpinfo() should display the mod_rewrite module as shown in Figure 7.13.

- We will be using the ".htaccess" file for defining our URL rewrite rules and therefore the web folder containing our PHP code will require the following configuration to be in place:
  The "**<Directory "/var/www/html">**" section within the main Apache Server configuration file **/etc/httpd/conf/httpd.conf** should look something like

this as shown below, with the "**AllowOverride All**" set for the **"/var/www/html"** folder. As this folder will contain our MVC web application:

```
# Further relax access to the default document root:
<Directory "/var/www/html">
#
# Possible values for the Options directive are "None",  # "All", or any combination of:
# Indexes Includes FollowSymLinks
#SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* ---
# "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and
# important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
```

```
#
Options Indexes FollowSymLinks
#
# AllowOverride controls what directives may be placed in # .htaccess files.
# It can be "All", "None", or any combination of the
# keywords:
# Options FileInfo AuthConfig Limit
#
AllowOverride All
#
# Controls who can get stuff from this server.
#
Require all granted
</Directory>
```

Our code will reside inside the folder **"/var/www/html/mymvc"**, and we will use the following directory structure to store our code files in.

```
/var/www/html/mymvc/
|
├── application
|       |
|       ├── config
|       |       └── config.php
|       |
|       ├── controller
```

```
│      │      ├──── books.php
│      │      └──── home.php
│      │
│      ├──── lib
│      │      ├──── application.php
│      │      └──── controller.php
│      │
│      ├──── model
│      │      └──── booksmodel.php
│      │
│      └──── view
│      ├──── books
│      │      └──── index.php
│      │
│      ├──── footer.php
│      ├──── header.php
│      │
│      └──── home
│             ├──── aboutus.php
│             ├──── index.php
│             └──── login.php
├──── .htaccess
```

```
└── index.php
```

7.6 BUILDING OUR MVC FRAMEWORK

To start building our MVC framework, we will navigate to the "**/var/www/html**" folder. This is the default **DocumentRoot** for Apache and also it requires root privileges. Therefore, we will use the "**su -**" command first and type in the root password in the terminal window, as shown below:

```
[fedorauser@kkhsou ~]$ su -
Password:
[root@kkhsou ~]#
[root@kkhsou ~]# cd /var/www/html/
[root@kkhsou html]#
```

Next, we need to create a folder within the DocumentRoot to contain our code.Let us name this folder as "mymvc":

```
 [root@kkhsou html]# mkdir mymvc
[root@kkhsou html]# cd mymvc
[root@kkhsou mymvc]#
```

We will create the directory structure, as shown in the previous section, to store our code files as we go along likewise:

CREATING THE .htaccess FILE

Create a "**.htaccess**" file within the **/var/www/html/mymvc/**" folder, with the contents as shown below. The purpose of this file is to enable a single point of entry into our MVC framework.

. htaccess

```
# Prevents issues with controller named "index" and having a #root index.php
# more here: http://httpd.apache.org/docs/2.2/content-#negotiation.html
Options -MultiViews
# Activates URL rewriting (like #vmou.ac.in/controller/action/1/2/3)
RewriteEngine On
# Disallows others to look directly into /public/ folder
Options -Indexes
# When using the script within a sub-folder, put this path here
# If your app is in the root of your web folder, then leave it commented out
RewriteBase /mymvc/
# General rewrite rules
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-l
RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```

CREATING THE application FOLDER

Now, create a folder named "**application**" within "**/var/www/html/mymvc/**". Then, within this "application" folder create a "**config**" folder and add the file "config.php" with the contents as shown below (Script 7.9):

Script 7.9

```
<?php // Script 7.98 - config.php
/* Configuration  Please refer to
http://php.net/manual/en/function.define.php*/
/* Configuration for: Error reporting Useful to show problems during
development. */

error_reporting(E_ALL);
```

```
ini_set("display_errors", 1);
/* Configuration for: Project URL Put your URL here */

define('URL', 'http://localhost/mymvc/');
/* Configuration for: Database Define your database credentials, database
type etc. */

define('DB_TYPE', 'mysql');
define('DB_HOST', '127.0.0.1');
define('DB_NAME', 'mymvc');
define('DB_USER', 'root');
define('DB_PASS', 'yourpassword');

?>
```

CREATING THE DATABASE

Create a MySQL database named "mymvc" and add the following tables to it, as shown below:

```
[root@kkhsou mymvc]# mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 2

Server version: 5.5.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other
names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> CREATE DATABASE IF NOT EXISTS `mymvc`;
mysql>
mysql> show databases;
+----------------------+
| Database             |
+----------------------+
| information_schema   |
| kkhsou                    |
| mymvc                     |
| mysql                     |
| performance_schema   |
| test                 |
+----------------------+
6 rows in set (0.00 sec)
mysql>
mysql> use mymvc
Reading table information for completion of table and column names You can turn
off this feature to get a quicker startup with –A
Database changed
mysql>
mysql> CREATE TABLE `mymvc`.`books` (
      `id` int(11) NOT NULL AUTO_INCREMENT,
      `title` text COLLATE utf8_unicode_ci NOT NULL,
      `author` text COLLATE utf8_unicode_ci NOT NULL,
      `publisher` text COLLATE utf8_unicode_ci NOT NULL,
    `year` text COLLATE utf8_unicode_ci NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `id` (`id`)
    ) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT
CHARSET=utf8 COLLATE=utf8_unicode_ci;
Query OK, 0 rows affected (0.10 sec)
mysql>
mysql> INSERT INTO `mymvc`.`books` (`id`, `title`, `author`, `publisher`,
`year`) VALUES
    (1, 'Title A', 'Author A', 'Publisher A', '2001'),
    (2, 'Title B', 'Author B', 'Publisher B', '2002'),
    (3, 'Title C', 'Author C', 'Publisher C', '2003'),
    (4, 'Title D', 'Author D', 'Publisher D', '2004'),
    (5, 'Title E', 'Author E', 'Publisher E', '2005'),
    (6, 'Title F', 'Author F', 'Publisher F', '2006'),
    (7, 'Title G', 'Author G', 'Publisher G', '2007'),
    (8, 'Title H', 'Author H', 'Publisher H', '2008'),
    (9, 'Title I', 'Author I', 'Publisher I', '2009'),
    (10, 'Title J', 'Author J', 'Publisher J', '2010'),
    (11, 'Title K', 'Author K', 'Publisher K', '2011'),
    (12, 'Title L', 'Author L', 'Publisher L', '2012');
Query OK, 12 rows affected (0.04 sec)
Records: 12 Duplicates: 0 Warnings: 0
mysql>
mysql> select * from books;
```

```
+----+---------+----------+------------+------+
| id | title   | author   | publisher  | year |
+----+---------+----------+------------+------+
| 1  | Title A | Author A | Publisher A | 2001 |
| 2  | Title B | Author B | Publisher B | 2002 |
| 3  | Title C | Author C | Publisher C | 2003 |
| 4  | Title D | Author D | Publisher D | 2004 |
| 5  | Title E | Author E | Publisher E | 2005 |
| 6  | Title F | Author F | Publisher F | 2006 |
| 7  | Title G | Author G | Publisher G | 2007 |
| 8  | Title H | Author H | Publisher H | 2008 |
| 9  | Title I | Author I | Publisher I | 2009 |
| 10 | Title J | Author J | Publisher J | 2010 |
| 11 | Title K | Author K | Publisher K | 2011 |
| 12 | Title L | Author L | Publisher L | 2012 |
+----+---------+----------+------------+------+
12 rows in set (0.00 sec)
mysql>
mysql> CREATE TABLE `mymvc`.`users` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `username` varchar(32) NOT NULL,
    `password` varchar(32) NOT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `id` (`id`)
    )      ENGINE=InnoDB      AUTO_INCREMENT=31      DEFAULT
    CHARSET=utf8 COLLATE=utf8_unicode_ci;
Query OK, 0 rows affected (0.07 sec)
mysql>
mysql> INSERT INTO `mymvc`.`users` (`id`, `username`,
    password`) VALUES (1, 'kkhsou', md5('"abcd1234" .
    kKh|+-|SoU%'"));
Query OK, 1 row affected (0.05 sec)
mysql>
mysql> select * from users;
+----+----------+----------------------------------+
| id  | username    | password                                  |
+----+----------+----------------------------------+
| 1   | vmou        | 76133bf68dd9d2bbba9f36e4498c92bf |
+----+----------+----------------------------------+
1 row in set (0.00 sec)
mysql>
```

Alternatively, we could also upload the following SQL file from the MySQL prompt, by typing the command "mysql -u root -p < database-tables.sql".

database-tables.sql

```
CREATE DATABASE IF NOT EXISTS `mymvc`;
CREATE TABLE `mymvc`.`books` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`title` text COLLATE utf8_unicode_ci NOT NULL,
`author` text COLLATE utf8_unicode_ci NOT NULL,
`publisher` text COLLATE utf8_unicode_ci NOT NULL,
`year` text COLLATE utf8_unicode_ci NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
CREATE TABLE `mymvc`.`users` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`username` varchar(32) NOT NULL,
`password` varchar(32) NOT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=31 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;
INSERT INTO `mymvc`.`books` (`id`, `title`, `author`, `publisher`, `year`)
VALUES
(1, 'Title A', 'Author A', 'Publisher A', '2001'),
(2, 'Title B', 'Author B', 'Publisher B', '2002'),
(3, 'Title C', 'Author C', 'Publisher C', '2003'),
(4, 'Title D', 'Author D', 'Publisher D', '2004'),
(5, 'Title E', 'Author E', 'Publisher E', '2005'),
(6, 'Title F', 'Author F', 'Publisher F', '2006'),
(7, 'Title G', 'Author G', 'Publisher G', '2007'),
(8, 'Title H', 'Author H', 'Publisher H', '2008'),
(9, 'Title I', 'Author I', 'Publisher I', '2009'),
(10, 'Title J', 'Author J', 'Publisher J', '2010'),
(11, 'Title K', 'Author K', 'Publisher K', '2011'),
(12, 'Title L', 'Author L', 'Publisher L', '2012');
INSERT INTO `mymvc`.`users` (`id`, `username`, `password`) VALUES (1,
'vmou',
md5('"abcd1234" . "@vm|+-|oU%"'));
```

CREATING THE index.php FILE

Next, we create an "**index.php**" file with the contents shown below, inside the folder "**/var/www/html/mymvc/**". This file will load all our configurations and classes (Script 7.10).

Script 7.10

```
<?php // Script 7.10 - index.php
/* A simple MVC skeleton */
// load application config (error reporting etc.)

require 'application/config/config.php';
// load application class

require 'application/lib/application.php';
require 'application/lib/controller.php';

// start the application
$app = new Application();

?>
```

CREATING THE DEFAULT CLASSES

Create the application and controller classes and put them in the **“/var/www/html/mymvc/application/lib/”** folder. For this we create two separate files **“application.php”** (Script 7.11) and **“controller.php”** (Script 7.12), and populate them with the respective contents shown below:

Script 7.11

```
<?php // Script 7.11 application.php

class Application
{
/* The controller set to null */
private $url_controller = null;
/* The method (of the above controller) set to null */
private $url_action = null;
/* Set the Parameter one to null */
private $url_parameter_1 = null;
/* Set the Parameter two to null */
 private $url_parameter_2 = null;
/* Set the Parameter three to null */
private $url_parameter_3 = null;
 /* "Start" the application:  * Analyzes the URL elements and calls
accordingly
 * the controller/method or the fallback */
public function __construct()
{
```

```
// create array with URL parts in $url
$this->splitUrl();
// check controller: does such a controller exist ?
if  (file_exists('./application/controller/'  .  $this->url_controller  .
'.php')) {
// if so, then load this file and create this controller
// example: if controller would be "login", then this line
// would translate into: $this->login = new login();
 require './application/controller/' . $this->url_controller . '.php';
$this->url_controller = new $this->url_controller();
// check method: does such a method exist in the controller ?
if (method_exists($this->url_controller, $this->url_action)) {
// call the method and pass the arguments to it
if (isset($this->url_parameter_3)) {
// will translate to something like $this->home->method($param_1,
$param_2, $param_3);
$this->url_controller->{$this->url_action}($this->url_parameter_1,
$this->url_parameter_2, $this->url_parameter_3);
} elseif (isset($this->url_parameter_2)) {
// will translate to something like $this->home->method($param_1,
$param_2);
$this->url_controller->{$this->url_action}($this->url_parameter_1,
$this->url_parameter_2);
} elseif (isset($this->url_parameter_1)) {
// will translate to something like $this->home->method($param_1);
```

```
                $this->url_controller->{$this->url_action}($this->url_parameter_1);
                } else {
                // if no parameters given, just call the method without parameters,
                like
                $this->home->method();
                $this->url_controller->{$this->url_action}();
                 }
                 } else {
                // default/fallback: call the index() method of a selected controller
                $this->url_controller->index();
                }
                } else {
    // invalid URL, so simply show home/index
    require './application/controller/home.php';
$home = new Home();
$home->index();
}
}
/* Get and split the URL */
private function splitUrl()
    {
    if (isset($_GET['url'])) {
// split URL
$url = rtrim($_GET['url'], '/');
$url = filter_var($url, FILTER_SANITIZE_URL);
$url = explode('/', $url);
// Put URL parts into according properties
// By the way, the syntax here is just a short form of if/else, called "Ternary
Operators"
$this->url_controller = (isset($url[0]) ? $url[0] : null);
$this->url_action = (isset($url[1]) ? $url[1] : null);
 $this->url_parameter_1 = (isset($url[2]) ? $url[2] : null);
 $this->url_parameter_2 = (isset($url[3]) ? $url[3] : null);
 $this->url_parameter_3 = (isset($url[4]) ? $url[4] : null;
 // for debugging. uncomment this if you have problems with the URL
// echo 'Controller: ' . $this->url_controller . '<br />';
// echo 'Action: ' . $this->url_action . '<br />';
// echo 'Parameter 1: ' . $this->url_parameter_1 . '<br />';
// echo 'Parameter 2: ' . $this->url_parameter_2 . '<br />';
// echo 'Parameter 3: ' . $this->url_parameter_3 . '<br />';
}
}
```

Script 7.12

```
<?php // Script 7.12 - controller.php
<?php
/* This is the "base controller class".
* All other "real" controllers extend this class.*/

class Controller
{
 /*The Database Connection is set to null */
 public $db = null;
/* Whenever a controller is created, open
 * a database connection too. The idea behind
```

```
                * this is to have ONE connection that can be
               * used by multiple models (there are
               * frameworks that open one connection per
               * model). */

               function __construct()
                      {
               $this->openDatabaseConnection();
               }
       /*Open the database connection with the
       *credentials from the
       *application/config/config.php file
               */
       private function openDatabaseConnection()
        {
       // set the (optional) options of the PDO connection.
        // In this case, we set the fetch mode to "objects",
        // which means all results will be objects, like this:
        // $result->username
        // For example, the fetch mode FETCH_ASSOC would return
        // results like this: $result["username"]
        // see http://www.php.net/manual/en/pdostatement.fetch.php
        $options = array(PDO::ATTR_DEFAULT_FETCH_MODE =>
       PDO::FETCH_OBJ, PDO::ATTR_ERRMODE =>
       PDO::ERRMODE_WARNING);
        // generate a database connection and using the PDO connector
        $this->db = new PDO(DB_TYPE . ':host=' . DB_HOST . ';dbname=' .
       DB_NAME, DB_USER, DB_PASS,
    $options);
    }
    /* Load the model with the given name.
     * loadModel("BooksModel") would include models/booksmodel.php
    * and create the object in the controller, like this:
     * $books_model = $this->loadModel('BooksModel');
    * Note that the model class name is written in "CamelCase",
    * the model's filename is the same in lowercase letters
     * param string $model_name The name of the model
    * return object model
            */
             public function loadModel($model_name)
            {
             require 'application/model/' . strtolower($model_name) . '.php';
             // return new model (and pass the database connection to the model)
             return new $model_name($this->db);
            }
            }
```

CREATING AND ADDING SOME CONTROLLERS

We have not yet created a controller and create a default controller called "home", which will also be our default application page with file named "home.php" (Script 7.13) as shown below. This file will saved inside the folder "/var/www/html/mymvc/application/controller/".

Script 7.13

```
<?php // Script 7.13 home.php
/* Class Home Please note: Don't use the same name  for class and method,as
 * this might trigger an (unintended) __constructof the class. This is really
* weird behaviour, *butdocumented here:
*http://php.net/manual/en/language.oop5.decon.php */

class Home extends Controller
{
/* PAGE: index This method handles what happens when you move to
*http://yourproject/home/index (which is the default page) */
public function index()
{
// debug message to show where you are, just for the demo
echo 'Controller says: You are in the controller home, using the method
index()';
// load some views
require 'application/view/header.php';
require 'application/view/home/index.php';
require 'application/view/footer.php';
}
/* * PAGE: login This method handles what happens when you move to
* http://localhost/home/login  The camelCase writing is just for better
*readability.  The method name is case insensitive. */
public function logIn()
{
// debug message to show where you are, just for the demo
echo 'Controller says: You are in the controller home, using the method
logIn()';
// load some views
require 'application/view/header.php';
require 'application/view/home/login.php';
require 'application/view/footer.php';
}
/*
* PAGE: aboutus
* This method handles what happens when you move to
* http://localhost/home/aboutus
* The camelCase writing is just for better readability.
* The method name is case insensitive.
 */
 public function aboutUs()
 {
 // debug message to show where you are, just for the demo
echo 'Controller says: You are in the controller home, using the method
aboutUs()';
 // load some views
 require 'application/view/header.php';
 require 'application/view/home/aboutus.php';
 require 'application/view/footer.php';
 }
}
?>
```

Let us create another controller called "books" with file named "books.php" (Script 7.14), as shown below.
Script 7.14

```
<?php // Script 7.14 books.php
 /* Class Books Please note: Don't use the same name for class
* and method, as this might trigger an (unintended) __construct
* of the class. This is really weird behaviour, but documented here:
* http://php.net/manual/en/language.oop5.decon.php */
class Books extends Controller
{
/* PAGE: index This method handles what happens when you move to
* http://localhost/mymvc/books/index */
public function index()
{
// simple message to show where you are
echo 'Controller says: You are in the Controller <strong>Books</strong>,
using the method
index().';
// load a model, perform an action, pass the returned data to a variable
// NOTE: please write the name of the model "LikeThis"
$books_model = $this->loadModel('BooksModel');
$books = $books_model->getAllBooks();
// load views. Also, within the views we can echo out $books
require 'application/view/header.php';
require 'application/view/books/index.php';
require 'application/view/footer.php';
}
/*
* ACTION: addBook
* This method handles what happens when you move to
* http://localhost/mymvc/books/addbook
* IMPORTANT: This is not a normal page, it's an ACTION.
* This is where the "add a book" form on books/index
* directs the user after the form submit. This method
* handles all the POST data from the form and then redirects
* the user back to books/index via the last line: header(...)
* This is an example of how to handle a POST request.
*/
public function addBook()
{
// simple message to show where you are
echo 'Controller says: You are in the Controller <strong>Books</strong>,
using the method
addBook().';
// if we have POST data to create a new book entry
if (isset($_POST["submit_add_book"])) {
// load model, perform an action on the model
$books_model = $this->loadModel('BooksModel');
$books_model->addBook($_POST["title"], $_POST["author"],
$_POST["publisher"],
$_POST["year"]);
}
// where to go after the book has been added
header('location: ' . URL . 'books/index');
}
/* ACTION: deleteBook  This method handles what happens when you
move
* to http://localhost/mymvc/books/deletebook
* IMPORTANT: This is not a normal page, it's an ACTION
* This is where the "delete a book" button on books/index
* directs the user after the click. This method handles
* all the data from the GET request (in the URL) and then
* redirects the user back to books/index via the last line: header(...)
* This is an example of how to handle a GET request.
 * param int $book_id Id of the book to delete  */
public function deleteBook($book_id)
 {
 // simple message to show where you are
 echo 'Controller says: You are in the Controller <strong>Books</strong>,
using the method
deleteBook().';
 // if we have an id of a book that should be deleted
 if (isset($book_id)) {
 // load model, perform an action on the model
 $books_model = $this->loadModel('BooksModel');
 $books_model->deleteBook($book_id);
 }
 // where to go after the book has been deleted
 header('location: ' . URL . 'books/index');
 }
}
?>
```

CREATING AND ADDING SOME VIEWS

Now, we will create the views in the "**/var/www/html/mymvc/application/view/**" folder. The view files, "**header.php**" (Script 7.15) and "**footer.php**" within this directory. The view files, "**index.php**", "**login.php**" and "**about.us**" within the subfolder "**home**" as these views are for the home controller.

Script 7.15

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
 <title>MVC skeleton</title>
<meta name="description" content="MVC skeleton">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<!-- css -->
<link href="<?php echo URL; ?>public/css/style.css" rel="stylesheet">
<!-- jQuery -->
<script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
<!-- our JavaScript -->
<script src="<?php echo URL; ?>public/js/application.js"></script>
</head>
<body>
<!-- header --> <div class="container">  <!-- Info -->
<div class="where-are-we-box">
Everything in this box is loaded from <span
class="bold">application/view/header.php</span> <br />
The green line is added via JavaScript (to show how to integrate JavaScript).
</div>
<h1>The header (used on all pages)</h1>
<!-- demo image -->
 <h3>Demo image, to show usage of public/img folder</h3>
 <div> <img src="<?php echo URL; ?>public/img/image.png" /> </div>
<!-- navigation -->
<h3>Demo Navigation</h3>
<div class="navigation">
<ul>
```

```
<!-- same like "home" or "home/index" -->
<li><a href="<?php echo URL; ?>"><?php echo URL; ?>home</a></li>
<li><a href="<?php echo URL; ?>home/login">
<?php echo URL; ?>home/login</a></li>
<li><a href="<?php echo URL; ?>home/aboutus"><?php echo URL;
?>home/aboutus</a></li> </ul></div>
<h3>Demo JavaScript</h3>
<div id="javascript-header-demo-box">
</div>
</div>
```

The index.php with the contents as shown below (Script 7.16), is within this folder containing the view for the controller named books:

Script 7.16

```
<div class="container">
 <h2>You are in the View: application/views/song/index.php</h2>
 <!-- add book form -->
 <div>
 <h3>Add a Book</h3>
 <form action="<?php echo URL; ?>books/addbook" method="POST">
 <label>Title</label>
 <input type="text" name="title" value="" required />
 <label>Author</label>
 <input type="text" name="author" value="" required />
 <label>Publisher</label>
 <input type="text" name="publisher" value="" />
 <label>Year</label>
```

```
<input type="text" name="year" value="" />
<input type="submit" name="submit_add_book" value="Submit" />
</form> </div>
<!-- main content output -->
<div> <h3>List of books (data from our first model)</h3>
<table> <thead style="background-color: #ddd; font-weight: bold;">
<tr>
<td>Book id</td>      <td>Book Title</td>   <td>Author</td>
<td>Publisher</td>
<td>Year</td>   <td>&nbsp;</td>   </tr>   </thead>
<tbody>
<?php foreach ($books as $book) { ?>
<tr>
<td><?php if (isset($book->id)) echo $book->id; ?></td>
<td><?php if (isset($book->title)) echo $book->title; ?></td>
<td><?php if (isset($book->author)) echo $book->author; ?></td>
<td><?php if (isset($book->publisher)) echo $book->publisher; ?></td>
<td><?php if (isset($book->year)) echo $book->year; ?></td>
<td><a href="<?php echo URL . 'books/deletebook/' . $book->id; ?>">DELETE
BOOK</a></td>
</tr>  <?php } ?>
</tbody>  </table>  </div>
</div>
```

We can also decide to put all our ".css" ".js" and image files, that will be used in our views, to reside within the "**/var/www/html/mymvc/view/**" folder. That way we know where all our view related files are located.

CREATING AND ADDING A MODEL

Now, we will create a model, which will perform some the database functions for our controller named books. Let us name our model file "booksmodel.php" (Script 7.17), populate it with the contents as shown below and save it within the folder "**/var/www/html/mymvc/application/model/**".

Script 7.17

```
<?php // Script 7.17 booksmodel.php
/* Class Books Please note:
*Don't use the same name for class and method,
as this might trigger an (unintended) __construct
of the class.
This is really weird behaviour, but documented here:
http://php.net/manual/en/language.oop5.decon.php */
class Books extends Controller
{
/* PAGE: index This method handles what happens when
you move to http://localhost/mymvc/books/index  */
public function index()
{
// simple message to show where you are
echo 'Controller says: You are in the Controller <strong>Books</strong>, using
the method
index().';
// load a model, perform an action, pass the returned data to a variable
// NOTE: please write the name of the model "LikeThis"
$books_model = $this->loadModel('BooksModel');
$books = $books_model->getAllBooks();
 // load views. Also, within the views we can echo out $books
```

```
    require 'application/view/header.php';
    require 'application/view/books/index.php';
    require 'application/view/footer.php';
    }
    /* ACTION: addBook This method handles what happens
   * when you move to http://localhost/mymvc/books/addbook
    * IMPORTANT: This is not a normal page, it's an ACTION.
    * This is where the "add a book" form on books/index
    * directs the user after the form submit. This method
    * handles all the POST data from the form and then redirects
    * the user back to books/index via the last line: header(...)
    * This is an example of how to handle a POST request.  */
    public function addBook()
    {
    // simple message to show where you are
    echo 'Controller says: You are in the Controller
   <strong>Books</strong>, using  the method
   addBook().';
    // if we have POST data to create a new book entry
    if (isset($_POST["submit_add_book"])) {
    // load model, perform an action on the model
    $books_model = $this->loadModel('BooksModel');
    $books_model->addBook($_POST["title"], $_POST["author"],
   $_POST["publisher"],
   $_POST["year"]);
    }
    // where to go after the book has been added
    header('location: ' . URL . 'books/index');
```

```
}
/*
* ACTION: deleteBook  this method handles what happens
*  when you move to http://localhost/mymvc/books/deletebook
* IMPORTANT: This is not a normal page, it's an ACTION.
* This is where the "delete a book" button on books/index
* directs the user after the click. This method handles
* all the data from the GET request (in the URL) and then
* redirects the user back to books/index via the last line: header(...)
* This is an example of how to handle a GET request.
* param int $book_id Id of the book to delete
 */
public function deleteBook($book_id)
{
// simple message to show where you are
echo 'Controller says: You are in the Controller <strong>Books</strong>,
using
the method
deleteBook().';
// if we have an id of a book that should be deleted
if (isset($book_id)) {
// load model, perform an action on the model
$books_model = $this->loadModel('BooksModel');
$books_model->deleteBook($book_id);
}
// where to go after the book has been deleted
header('location: ' . URL . 'books/index');
}
```

```
}
?>
```

7.7 Building a PHP Application on our MVC framework

In the previous section, we have built our PHP application. However, we will make some modifications and additions to the MVC framework in that application. Most of the PHP code contains comments for the ease of code readability and therefore we will go ahead and describe the final directory and file layout of the MVC application and list all the code files with the respective contents in this section.

THE FINAL DIRECTORY/FILE TREE

/var/www/html/mymvc/

```
├── application
│   ├── config
│   │   └── config.php
│   ├── controller
│   │   ├── books.php
│   │   └── home.php
│   ├── lib
│   │   ├── application.php
│   │   └── controller.php
│   ├── model
│   │   ├── booksmodel.php
│   │   └── loginmodel.php
│   └── view
│   ├── books
│   │   └── index.php
│   ├── css
│   │   └── mystyle.css
│   │
│   ├── footer.php
│   ├── header.php
│   ├── home
│   │   ├── aboutus.php
│   │   ├── index.php
│   │   └── login.php
│   ├── images
│   │   └── bg.gif
│   └── js
│   └── myjavascripts.js
├── .htaccess
└── index.php
```

THE .htaccess FILE

```
# To prevent problems when using a controller named
# "index" and having a root index.php
# more here: http://httpd.apache.org/docs/2.2/content-
#negotiation.html
Options -MultiViews
# Activates URL rewriting (like
# localhost/controller/action/1/2/3)
RewriteEngine On
# Disallows others to look directly into /folder/
Options -Indexes
# When using the script within a sub-folder, put this
# path here, like /mysubfolder/
# If your app is in the root of your web folder, then
# leave it commented out
RewriteBase /mymvc/
```

```
# General rewrite rules
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-l
RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```

CONTENTS OF THE index.php FILE

Script 7.18

```
<?php // Script 7.18 index.php
// start the session
session_start();
// set time-out period (in seconds)
$inactive = 60;
// check to see if $_SESSION["timeout"] is set
if (isset($_SESSION["timeout"])) {
 // calculate the session's "time to live"
 $sessionTTL = time() - $_SESSION["timeout"];
if ($sessionTTL > $inactive) {
 session_destroy();
header('location: ' . URL . 'home/logmeout');
 }
}
$_SESSION["timeout"] = time();
/*
* A simple MVC skeleton
*/
// load application config (error reporting etc.)
require 'application/config/config.php';
// load application class
```

```
require 'application/lib/application.php';
require 'application/lib/controller.php';
// start the application
$app = new Application();
?>
```

CONTENTS OF application/config/config.php FILE
Script 7.19

```
<?php // Script 7.19 config.php
/*
Configuration
*
See http://php.net/manual/en/function.define.php
*/
/*
Configuration for: Error reporting
Useful to show every little problem during development
*/
error_reporting(E_ALL);
ini_set("display_errors", 1);
/*
Configuration for: Project URL
Put your URL here, "127.0.0.1" or "localhost"
*/
define('URL', 'http://localhost/mymvc/');
/*
Configuration for: Database
Define your database credentials, database type etc.
*/
```

```
define('DB_TYPE', 'mysql');
define('DB_HOST', '127.0.0.1');
define('DB_NAME', 'mymvc');
define('DB_USER', 'root');
define('DB_PASS', 'dbpassword');
/*
* Salt to enhance password security
*/
define('MY_SALT', '@vm|+-|oU%');
?>
```

CONTENTS OF application/controller/books.php FILE

Script 7.20

```
<?php // Script 7.20 books.php
/*
Class Books
*
Please note:
Don't use the same name for class and method,
as this might trigger an (unintended) __construct
of the class.
This is really weird behaviour, but documented here:
http://php.net/manual/en/language.oop5.decon.php
*
*/
class Books extends Controller
{
/*
```

```
PAGE: index
This method handles what happens when you move to
http://localhost/mymvc/books/index
*/
public function index()
{
// simple message to show where you are
echo 'Controller says: You are in the Controller
<strong>Books</strong>, using the method index().';
// load a model, perform an action, pass the returned data to a
variable
// NOTE: please write the name of the model "LikeThis"
$books_model = $this->loadModel('BooksModel');
$books = $books_model->getAllBooks();
// load views. Also, within the views we can echo out $books
require 'application/view/header.php';
require 'application/view/books/index.php';
require 'application/view/footer.php';
}
/*
ACTION: addBook
This method handles what happens when you move to
http://localhost/mymvc/books/addbook
IMPORTANT: This is not a normal page, it's an ACTION.
This is where the "add a book" form on books/index
directs the user after the form submit. This method
handles all the POST data from the form and then redirects
the user back to books/index via the last line: header(...)
```

```
This is an example of how to handle a POST request.
*/
public function addBook()
{
// simple message to show where you are
echo 'Controller says: You are in the Controller
<strong>Books</strong>, using the method addBook().';
// if we have POST data to create a new book entry
if (isset($_POST["submit_add_book"])) {
// load model, perform an action on the model
$books_model = $this->loadModel('BooksModel');
$books_model->addBook($_POST["title"], $_POST["author"],
$_POST["publisher"], $_POST["year"]);
}
// where to go after the book has been added
header('location: ' . URL . 'books/index');
}
/*
ACTION: deleteBook
This method handles what happens when you move to
http://localhost/mymvc/books/deletebook
IMPORTANT: This is not a normal page, it's an ACTION.
This is where the "delete a book" button on books/index
directs the user after the click. This method handles
all the data from the GET request (in the URL) and then
redirects the user back to books/index via the last line: header(...)
This is an example of how to handle a GET request.
param int $book_id Id of the book to delete
```

```
*/
public function deleteBook($book_id)
{
// simple message to show where you are
echo 'Controller says: You are in the Controller
<strong>Books</strong>, using the method deleteBook().';
// if we have an id of a book that should be deleted
if (isset($book_id)) {
// load model, perform an action on the model
$books_model = $this->loadModel('BooksModel');
$books_model->deleteBook($book_id);
}
// where to go after the book has been deleted
header('location: ' . URL . 'books/index');
}
}
?>
```

CONTENTS OF application/controller/home.php FILE

Script 7.21

```
<?php // Script 7.21 home.php
/*
* Class Home
*
* Please note:
* Don't use the same name for class and method,
* as this might trigger an (unintended) __construct
* of the class. This is really weird behaviour, but
* documented here:
* http://php.net/manual/en/language.oop5.decon.php
*
*/
class Home extends Controller
{
 /*
 * PAGE: index
 * This method handles what happens when you move
 * to http://yourproject/home/index (which is the
* default page)
 */
 public function index()
 {
 // debug message to show where you are, just for the demo
 echo 'Controller says: you are in the controller
<strong>home</strong>, using the method <em>index()</em>.';
 // load some views
 require 'application/view/header.php';
 require 'application/view/home/index.php';
 require 'application/view/footer.php';
 }
 /*
 * PAGE: login
 * This method handles what happens when you move to
 * http://localhost/home/login
 * The camelCase writing is just for better readability.
 * The method name is case insensitive.
```

```
*/
public function logIn()
{
// debug message to show where you are, just for the demo
echo 'Controller says: you are in the controller
<strong>home</strong>, using the method <em>login()</em>.';
// load some views
require 'application/view/header.php';
require 'application/view/home/login.php';
require 'application/view/footer.php';
}
/*
* ACTION: logmein
* This method handles what happens when you move to
* http://localhost/mymvc/home/logmein
* IMPORTANT: This is not a normal page, it's an ACTION.
* This is where the "Login" button on the Login Form
* directs the user after the login form submit. This method
* handles all the POST data from the form and then redirects
* the user to books/index via the last line: header(...)
*/
public function logMeIn()
{
// simple message to show where you are
echo 'Controller says: you are in the Controller
<strong>home</strong>, using the method <em>logmein()</em>.';
// if we have POST data to login
if (isset($_POST["submit_login"])) {
```

```
// load model, perform an action on the model
 $login_model = $this->loadModel('LoginModel');
 // check username and password combination
 if ($login_model->checkUser($_POST["username"],
$_POST["password"])) {
 // set the session variables
 $_SESSION["username"] = $_POST["username"];
 $_SESSION["loggedin"] = TRUE;
// username password correct
 header('location: ' . URL . 'books/index');
 } else {
// username password incorrect
echo '<p><font color="#ff0000">Please check your username and
password and try again</font></p>';
 // load some views
 require 'application/view/header.php';
 require 'application/view/home/login.php';
 require 'application/view/footer.php';
 }
 }
 }
 /*  ACTION: logmeout
 * This method handles what happens when you move to
 * http://localhost/mymvc/home/logmeout
 * IMPORTANT: This is not a normal page, it's an ACTION.
 * This is where the "Logout" button on the Login Page
 * directs the user to the login form page. This method
 * handles resets all the session variables and then redirects
 * the user to home/index via the last line: header(...)      */
 public function logMeOut()
 {
 // simple message to show where you are
 echo 'Controller says: you are in the Controller
<strong>home</strong>, using the method <em>logmeout()</em>.';
// delete the username, loggedin values
unset($_SESSION["username"]);
unset($_SESSION["loggedin"]);
// unset all session values
session_unset();
 // destroy the session
 session_destroy();
// redirect to the default index page
 header('location: ' . URL . 'home/index');
}
 /*
 * PAGE: aboutus
 * This method handles what happens when you move to
 * http://localhost/home/aboutus
 * The camelCase writing is just for better readability.
 * The method name is case insensitive.
 */
 public function aboutUs()
 {
 // debug message to show where you are, just for the demo
 echo 'Controller says: you are in the controller
<strong>home</strong>, using the method <em>aboutus()</em>.';
```

```
 // load some views
 require 'application/view/header.php';
 require 'application/view/home/aboutus.php';
 require 'application/view/footer.php';
}       }
?>
```

CONTENTS OF THE application/lib/application.php

Script 7.22

```
<?php //Script 7.22 application.php
class Application
{
 /* The controller set to null */
 private $url_controller = null;
 /* The method (of the above controller) set to null */
 private $url_action = null;
 /* Set the Parameter one to null */
 private $url_parameter_1 = null;
 /* Set the Parameter two to null */
 private $url_parameter_2 = null;
 /* Set the Parameter three to null */
 private $url_parameter_3 = null;
/*
 * "Start" the application:
 * Analyzes the URL elements and calls accordingly
 * the controller/method or the fallback
 */
 public function __construct()
 {
 // create array with URL parts in $url
$this->splitUrl();
 // check controller: does such a controller exist ?
 if (file_exists('./application/controller/' . $this->url_controller .
'.php')) {
 // if so, then load this file and create this controller
 // example: if controller would be "login", then this line
 // would translate into: $this->login = new login();
 require './application/controller/' . $this->url_controller .
'.php';
 $this->url_controller = new $this->url_controller();
 // check method: does such a method exist in the controller ?
 if (method_exists($this->url_controller, $this->url_action)) {
 // call the method and pass the arguments to it
if (isset($this->url_parameter_3)) {
 // will translate to something like $this->home-
>method($param_1, $param_2, $param_3);
 $this->url_controller->{$this->url_action}($this-
>url_parameter_1, $this->url_parameter_2, $this->url_parameter_3);
 } elseif (isset($this->url_parameter_2)) {
 // will translate to something like $this->home-
>method($param_1, $param_2);
 $this->url_controller->{$this->url_action}($this-
>url_parameter_1, $this->url_parameter_2);
 } elseif (isset($this->url_parameter_1)) {
 // will translate to something like $this->home-
>method($param_1);
```

```
 $this->url_controller->{$this->url_action}($this-
>url_parameter_1);
 } else {
 // if no parameters given, just call the method without
parameters, like $this->home->method();
 $this->url_controller->{$this->url_action}();
 }
 } else {
 // default/fallback: call the index() method of a selected
Controller
$this->url_controller->index();
                    }
 } else {
 // invalid URL, so simply show home/index
 require './application/controller/home.php';
$home = new Home();
 $home->index();
 }
 }
 /*
 * Get and split the URL
 */
 private function splitUrl()
 {
 if (isset($_GET['url'])) {
 // split URL
 $url = rtrim($_GET['url'], '/');
 $url = filter_var($url, FILTER_SANITIZE_URL);
 $url = explode('/', $url);
 // Put URL parts into according properties
 // By the way, the syntax here is just a short form of if/else,
called "Ternary Operators"
 // @see http://davidwalsh.name/php-shorthand-if-else-ternaryoperators
 $this->url_controller = (isset($url[0]) ? $url[0] : null);
 $this->url_action = (isset($url[1]) ? $url[1] : null);
 $this->url_parameter_1 = (isset($url[2]) ? $url[2] : null);
 $this->url_parameter_2 = (isset($url[3]) ? $url[3] : null);
$this->url_parameter_3 = (isset($url[4]) ? $url[4] : null);
 // for debugging. uncomment this if you have problems with the URL
 // echo 'Controller: ' . $this->url_controller . '<br />';
 // echo 'Action: ' . $this->url_action . '<br />';
 // echo 'Parameter 1: ' . $this->url_parameter_1 . '<br />';
 // echo 'Parameter 2: ' . $this->url_parameter_2 . '<br />';
 // echo 'Parameter 3: ' . $this->url_parameter_3 . '<br />';
 }
 }
}
?>
```

CONTENTS OF application/lib/controller.php File

Script 7.23

```
<?php //Script 7.23 controller.php
/*  This is the "base controller class".  Other "real" controllers extend this
class.  */
class Controller
{
 /* The Database Connection is set to null  */
```

```
public $db = null;
/* Whenever a controller is created, open a database connection
* too. The idea behind this is to have ONE connection that can be
 * used by multiple models (there are frameworks
 * that open one connection per model).  */
function __construct()
{
$this->openDatabaseConnection();
}
/*  Open the database connection with the credentials
 * from the application/config/config.php file    */
private function openDatabaseConnection()
{
// set the (optional) options of the PDO connection.
// In this case, we set the fetch mode to "objects",
// which means all results will be objects, like this:
// $result->username
// For example, the fetch mode FETCH_ASSOC would return
// results like this: $result["username"]
// see http://www.php.net/manual/en/pdostatement.fetch.php
$options = array(PDO::ATTR_DEFAULT_FETCH_MODE =>
PDO::FETCH_OBJ,
PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING);
// generate a database connection and using the PDO connector
$this->db = new PDO(DB_TYPE . ':host=' . DB_HOST . ';dbname=' .
DB_NAME, DB_USER, DB_PASS, $options);
}
/* Load the model with the given name.
```

```
 * loadModel("BooksModel") would include models/booksmodel.php
 * and create the object in the controller, like this:
 * $books_model = $this->loadModel('BooksModel');
 * Note that the model class name is written in "CamelCase",
 * the model's filename is the same in lowercase letters
 * param string $model_name The name of the model  return object model
*/
 public function loadModel($model_name)
 {
 require 'application/model/' . strtolower($model_name) . '.php';
 // return new model (and pass the database connection to the model)
 return new $model_name($this->db);
 }
}
?>
```

CONTENTS OF application/model/booksmodel.php FILE

Script 7.24

```
<?php //Script 7.24 booksmodel.php
<?php
class BooksModel
{
 /*
 * Every model needs a database connection, passed to the model
 * param object $db is a PDO database connection
 */
 function __construct($db) {
```

```
try {
$this->db = $db;
} catch (PDOException $e) {
exit('Database connection could not be established.');
}
}
/*
* Get all the books from the database
*/
public function getAllBooks()
{
$sql = "SELECT id, title, author, publisher, year FROM books";
$query = $this->db->prepare($sql);
$query->execute();
// fetchAll() is the PDO method that gets all result rows,
// here in object-style because we defined this in the file
// lib/controller.php. If you prefer to get an associative
// array as the result, then $query->fetchAll(PDO::FETCH_ASSOC);
// or change in the lib/controller.php's PDO options to
// $options = array(PDO::ATTR_DEFAULT_FETCH_MODE =>
PDO::FETCH_ASSOC
...
return $query->fetchAll();
}
/*
* Add a book to the database
* param string $title Title
* param string $author Author
```

```
 * param string $publisher Publisher
 * param string $year Year
 */
 public function addBook($title, $author, $publisher, $year)
 {
 // clean the input from javascript code for example
 $title = strip_tags($title);
$author = strip_tags($author);
 $publisher = strip_tags($publisher);
 $year = strip_tags($year);
 $sql = "INSERT INTO books (title, author, publisher, year) VALUES
(:title, :author, :publisher, :year)";
 $query = $this->db->prepare($sql);
 $query->execute(array(':title' => $title, ':author' => $author,
':publisher' => $publisher, ':year' => $year));
 }
 /*
 * Delete a book from the database
 * param int $book_id Id of a Book
 */
 public function deleteBook($book_id)
 {
 $sql = "DELETE FROM books WHERE id = :book_id";
 $query = $this->db->prepare($sql);
 $query->execute(array(':book_id' => $book_id));
 }
}
?>
```

CONTENTS OF application/model/loginmodel.php FILE

Script 7.25

```
<?php // Script 7.25loginmodel.php
class LoginModel
{
 /*
 * Every model needs a database connection, passed to the model
 * param object $db is a PDO database connection
 */
 function __construct($db) {
 try {
 $this->db = $db;
 } catch (PDOException $e) {
 exit('Database connection could not be established.');
 }
 }
 /*
 * Check the username and password in the database
 * param string $username Login Username
 * param string $password Login Password
 */
 public function checkUser($username, $password)
 {
 $username = strip_tags($username);
 $password = strip_tags($password);
 $sql = "SELECT username, password FROM `users` WHERE username =
'".$username."' AND password = md5('\"".$password."\" .
\"".MY_SALT."\"')";
```

```
 $query = $this->db->prepare($sql);
$query->execute();
$numrows = $query->rowCount();
 // fetchAll() is the PDO method that gets all result rows,
 // here in object-style because we defined this in the file
 // lib/controller.php. If you prefer to get an associative
 // array as the result, then $query->fetchAll(PDO::FETCH_ASSOC);
 // or change in the lib/controller.php's PDO options to
 // $options = array(PDO::ATTR_DEFAULT_FETCH_MODE =>
PDO::FETCH_ASSOC
...
 if ($numrows > 0) {
// getting to this point means that the credentials match
return TRUE;
} else {
// getting to this points means unsuccessful login
return FALSE;
}
 }
}
?>
```

CONTENTS OF application/view/books/index.php FILE

Script 7.26

```
<?php // Script 7.25 index.php
// check and see if the user is logged in
if (isset($_SESSION["loggedin"])) {
// user is logged in show the Page
echo '<p>Hey you have logged in with the username <strong>' .
```

```
$_SESSION["username"] . '</strong></p>';
?>
<div class="container">
<p>You are in the View:
"<strong>application/views/song/index.php</strong>"</p>
<!-- add book form -->
<div>
<h3>Add a Book</h3>
<form action="<?php echo URL; ?>books/addbook" method="POST">
<label>Title</label>
<input type="text" name="title" value="" required />
<label>Author</label>
<input type="text" name="author" value="" required />
<label>Publisher</label>
<input type="text" name="publisher" value="" required/>
<label>Year</label>
<input type="text" name="year" value="" required/>
<input type="submit" name="submit_add_book" value="Submit" />
</form>
</div>
<!-- main content output -->
<div>
<h3>List of books (data from our first model)</h3>
<table>
<thead style="background-color: #ddd; font-weight: bold;">
<tr>
<td>Book id</td>
<td>Book Title</td>
```

```
<td>Author</td>
<td>Publisher</td>
<td>Year</td>
<td>&nbsp;</td>
</tr>
</thead>                    <tbody>
<?php foreach ($books as $book) { ?>
<tr>
<td><?php if (isset($book->id)) echo $book->id; ?></td>
<td><?php if (isset($book->title)) echo $book->title;
?></td>
<td><?php if (isset($book->author)) echo $book->author;
?></td>
<td><?php if (isset($book->publisher)) echo $book-
>publisher; ?></td>
<td><?php if (isset($book->year)) echo $book->year;
?></td>
<td><a href="<?php echo URL . 'books/deletebook/' . $book-
>id; ?>">DELETE BOOK</a></td>
</tr>
<?php } ?>
</tbody>
</table>
</div>
</div>
<?php
} else {
// user is not logged in, send user to login page
```

```
header('location: ' . URL . 'home/login');
}
?>
```

CONTENTS OF application/view/css/mystyle.css FILE

```
/* login-form class definition */
.login-form {
color:darkred; /* darkred text colour */
text-transform:uppercase; /* text in uppercase */
border-style:dotted; /* dotted border */
border-width:2px; /* 2 pixels border width */
border-color:#ff0000; /* red border colour */
background-color:#cccccc; /* background colour is a shade of grey */
}
/* apply to all the td elements of the login-form class */
.login-form td {
text-align:center; /* center align td elements */
vertical-align:middle; /* vertically middle align td elements */
}
/* apply to all the input elements of the login-form class */
.login-form input {
color:darkred; /* input elements have darkred text colour */
background-color:#eeeeee; /* input elements have grey background */
height:25px; /* input elements have 25 pixels height */
width:200px; /* input elements have 200 pixels width */
}
.view-box{
height: 500px;
width: 500px;
position: relative;
background-color: #FFFFFF;
border-width: 1px;
border-style: solid;
border-color: #dddddd;
border-radius: 0px;
box-shadow: 0px 10px 6px -6px #777;
}
/* some styling for the menu */
ul.menu
{
padding:0;
list-style-type: none;
height: 26px;
/*width:500px;margin:0 auto;*//*Uncomment this line to make the menu
center-aligned.*/
}
```

CONTENTS OF application/view/header.php FILE

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>MVC skeleton</title>
<meta name="description" content="MVC skeleton">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<!-- css -->
<link href="<?php echo URL; ?>application/view/css/mystyle.css"
rel="stylesheet">
<!-- jQuery -->
<script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
<!-- our JavaScript -->
<script src="<?php echo URL;
?>application/view/js/myjavascripts.js"></script>
</head>
<body>
<!-- header -->
<div class="container">
<div class="header">
<p>This is my header</p>
</div>
<!-- navigation -->
<div class="navigation">
<ul class="menu">
<!-- same like "home" or "home/index" -->
<li><a href="<?php echo URL; ?>">Home</a></li>
<li><a href="<?php echo URL; ?>home/login">Login</a></li>
<li><a href="<?php echo URL; ?>home/aboutus">Aboutus</a></li>
<li class="lastItem"><a href="<?php echo URL;
?>home/logmeout">Logout</a></li>
</ul>
</div>
</div>
```

CONTENTS OF application/view/home/aboutus.php FILE

```
<div class="view-box">
<p>You are in the View:
"<strong>application/view/home/aboutus.php</strong>"</p>
<p>You are in <strong>About Us</strong></p>
</div>
CONTENTS OF THE application/view/home/index.php FILE
<div class="view-box">
<p>You are in the View:
"<strong>application/view/home/index.php</strong>"</p>
<p>You are in <strong>Home</strong></p>
</div>
CONTENTS OF THE application/view/home/login.php FILE
<div class="view-box">
<p>You are in the View:
"<strong>application/view/home/login.php</strong>"</p>
<p>You are in <strong>Login</strong></p>
<form action="<?php echo URL; ?>home/logmein" method="POST">
<table class="login-form">
<tr>
<td>Message:</td><td><input id="messageBox" name="messageBox"
type="text" name="messageBox" readonly></td>
</tr>
<tr>
<td>Username:</td><td><input id="username" name="username"
type="text" name="username" onkeyup="checkUsername()"></td>
</tr>
<tr>
```

```
<td>Password:</td><td><input id="password" name="password"
type="password" name="password"></td>
</tr>
<tr>
<td colspan="2"><input id="submit_login" name="submit_login"
type="submit" value="Login"></td>
</tr>
</table>
</form>
</div>
```

CONTENTS OF application/view/js/myjavascripts.js FILE

```
//function checkUsername checks the username input; whether blank,
//numbers, letters or special characters.
function checkUsername()
{
var username;
username = document.getElementById("username").value;
if ( username=="" )
{
document.getElementById("messageBox").value = "Blank Username";
}
else if ( username.match(/[a-z]/i) )
{
document.getElementById("messageBox").value = "Alphabets";
}
else if ( username.match(/[0-9]/) )
{
```

```
document.getElementById("messageBox").value = "Numbers";
}
else
{
document.getElementById("messageBox").value = "Special
Characters";
}
}
```

7.8 Self Learning Exercise

Q.1 MVC is

a) a web page

b) a scripting language

c) a framework

d) a markup language

Q.2 In MVC, what does the M stand for ?

a) Model

b) Markup

c) Modern

d) Machine

Q.3 What does the C stand for in MVC ?

a) C language

b) Characteristics

c) Controller

d) Compiler

Q.4 MVC stands for which of the following ?

a) Model View Controller

b) Modern View Compiler

c) Machine View Controller

d) Machine View Compiler

Q.5 Which of the following location contains the default DocumentRoot for the Apache Web Server?

a) /var/www/html

b) /var/log/httpd

c) /etc/httpd

d) /etc/www/html

7.9 Summary

In this unit, we have briefly discussed the basics of the MVC design pattern. Though we have covered only a few of the topics, these topics are intended to inspire and point us in a direction to further explore. This unit was not intended to provide student with an exhaustive in-depth on the topics but merely to introduce us to some of the basic concepts of the MVC architecture and use them in building web pages.

What we have learned in this unit.

- MVC basics and Coding considerations
- Building a MVC framework
- Using our MVC framework to write PHP a web application

7.10 Glossary

Web application - A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet.

Framework- a framework is often a layered structure indicating what kind of programs can or should be built and how they would interrelate.

MVC- Model-View-Controller (MVC) is the name of a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models.

Development Environment- In computer program and software product development, the development environment is the set of processes and programming tools used to create the program or software product.

7.11 Answers to Self-Learning Exercise

1. (c)
2. (a)
3. (c)
4. (a)
5. (a)

7.12 Exercise

Q.1 What is MVC ?

Q.2 How can the methods of a class be used in PHP?

Q.3 How do we test if the apache mod_rewrite module is available and working?

Q.4 Which command is used to create a database in MySQL?

Q.5 Which symbol/sign represents the start of a variable name in PHP?

Q.6 Which MySQL command is used to insert values in tables?

Q.7 Which Apache module is required for PHP scripts to connect to MySQL?

Q.8 What does the Apache mod_rewrite module do?

Q.9 Which are the start and end tags that the PHP parser reads to start parsing and stop parsing PHP code?

Q.10 Which command is used to list the table contents in MySQL?

References and Suggested Readings

1. Programming Microsoft ASP.NET MVC By Dino Esposito.
2. Pro PHP MVC By Chris Pitt, Apress,
3. ASP.NET MVC4, by Jess C., Todd S., Hrusikesh, O'Reilly

4. Java Coding Standards: Java Coding Guidelines: 75 Recommendations By Harry Hariom Choudhary, Sun.
5. Beginning PHP and MySQL: From Novice to Professional 4th Edition by W. Jason Gilmore, Apress
6. PHP: The Complete Reference 1st Edition, by Steven Holzner, Tata Mcgraw Hill Education Private Limited
7. http://www.php.net/manual/en/
8. http://en.wikipedia.org/wiki/PHP
9. http://in2.php.net/FAQ.php

UNIT-8
HTML 5

Structure of the Unit

8.0 Objective

After reading this chapter you will be able to understand the basics of HTML5 and the following:

- Elements are no longer supported by HTML5.
- A number of elements and attributes that can help in designing a modern webpages.
- Use of some semantics and API introduced in HTML5.

8.1 Introduction

HTML5 was introduced in year 2014 on the recommendations of W3C. The primary objective of HTML5 is to improve the language with support to latest multimedia. HTML5 have many new semantics for better represent on web like <header>, <footer>, <article> etc. HTML5 also introduce many new APIs/features like Geolocation, Drag and Drop, Local Storage etc.

8.2 Common Infrastructure

This section focus on some elements, tags, attributes that are common to the previous versions of HTML. The list of some common tags that are also works with HTML5:

There are also some elements that are deleted from HTML5. They are: <acronym>, <applet>, <basefont>, <big>, <center>, <dir>, <font>, <frame>, <frameset>, <noframes>. <strike>, and <tt>.

In the next section we are discussed the new features introduced in HTML5.

8.3 Semantics of HTML5

The word 'semantics' means study of the meaning of words and phrases in a paragraph. So the semantic elements are those which are associated with a meaning. HTML5 supports some semantic elements like <form>, <table>, <attitude> etc., which clearly describes the content in the elements. HTML5 introduces these semantic elements for better structure and easy for search engine to identify the content of web page. These elements are also supported by some modern web browsers like google- chrome, IE, Mozilla Firefox, Opera etc.

Previously many website uses HTML code like <div id="nav">, <div class= "header"> to represent navigation and header respectively. These HTML code is replaced with new semantic elements like <nav> and <header>. The following are the list of new semantics elements introduced in HTML5:

- <article>
- <aside>
- <details>
- <figcaption>

- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

HTML5 <section> element: The <section> element is used to define a section in a document of HTML5. For example: you may have introduction content and contact information each in a section element.

Example:

```
<section>
<h1> About VMOU<h1>
<p>VMOU is Vardhman Mahaveer Open University in Kota, Rajasthan</p>
</section>
```

HTML5 <article> element: HTML5 <article> element represents independent, self-contained content which reads independent from the rest of the website. For example: it can be used in Form post, Blog post, Newspaper article etc.

Example:

```
<article>
<h1> What does VMOU do? </h1>
<p> -----</p>
</article>
```

NOTE: We cannot nest <section> and <article> element.

HTML5 <header > element: It represent header for a document or section. There can be many headers in a document.

Example:

```
<article>
```

```
        <header>
        <h1> VMOU</h1>
        <p> An Open University</p>
         </header>
    <p>----------</p>
</article>
```

HTML5 <footer> Element: <footer> represents footer of the document or section. <footer> usually contains information about author, copyright information, contact information etc.

Example:

```
<footer>
<p>copyright by VMOU-2016</p>
<p>For more information: <a href=”www.vmou.ac.in>”click here</a
<p>e-mail us: <a href=”mailto: support@vmou.ac.in”> click here</a>
<footer>
```

HTML5 <nav> Element: This element contains the navigation links for the website.

Example:

```
<nav>
<a href=”www.vmou.ac.in”> Home </a>
<a href=”www.vmou.ac.in/exam”> Examination </a>
<a href=”www.vmou.ac.in/about” > About us </a>
</nav>
```

HTML5 <aside> element: HTML5<aside> element represents the content that are slightly related to rest of the content. This is usually when some contents are aside from the remaining content like the use of sidebar.

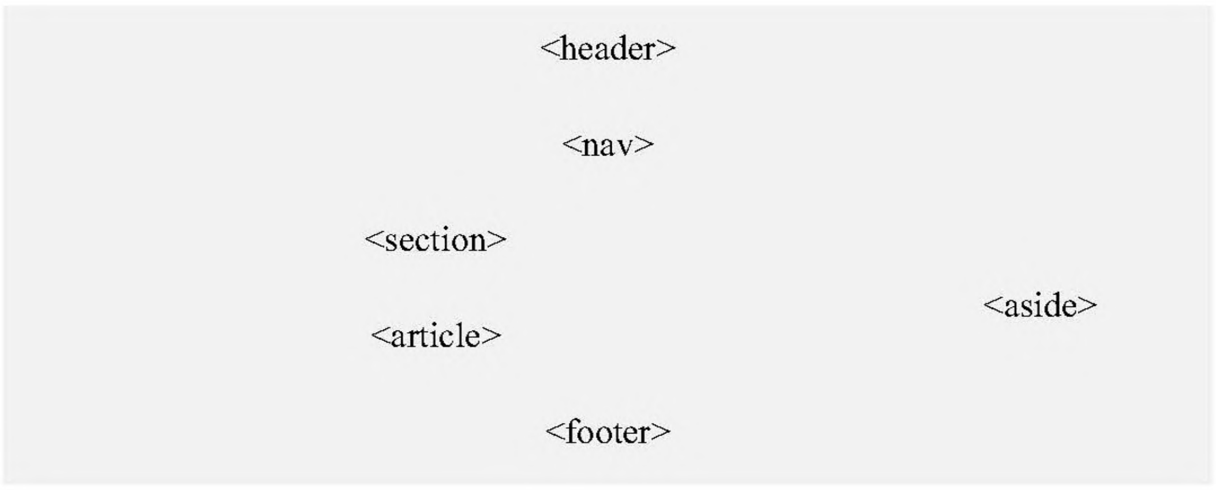
Example:

```
<aside>
  <p>This is aside content from the remaining part</p>
<aside>
```

HTML5<figure> and <figcaption> element: These tags are used to add a visual explanation to an image.

Example:

```
<figure>
<img src= "a.jpg">
<figcaption>Figure 1: First image</figcaption>
</figure>
```



There are more semantics elements present in HTML5. There brief description are shown in the following table:

S.No.	Semantic element	Description
1	<details>	Specifies the addition detail that can be hide or show by the user.
2	<main>	Specifies main content of the document/section.

3	<more>	Specifies highlighted texts.
4	<summary>	Visible heading for a <detail> element.
5	<time>	Specifies date/time.

The purpose of introducing semantic elements like <header>,<footer> etc. in HTML5 is to made it possible for a search engine to clearly identifies the content of the document. With the previous version of HTML, it is difficult to identify the content of the document by the search engine , as the developer uses id/class names to specify the elements like menu, footer etc.

8.4 HTML5 form tag

HTML5 has an improvement to its HTML classical form by introducing various new attributes, elements and input types. In this section we will study about these improvements.

The list of new form elements in HTML5:

- <datalist>
- <keygen>
- <output>

<datalist> element: This is a predefined list. You may select one of the values from the list.

Example:

```
<form action =”myaction.php”>
<input list =”university”>
<datalist id=” university”>
<option value=”Vardhman Mahaveer Open University”>
<option value=”Kota University”>
<option value=”Rajasthan Technical University”>
</datalist>
```

```
</form>
```

<keygen> element: This element defines a key-pair generator field (for forms).When the form is submitted the private key is stored locally and the public key is send to the source. <keygen>element specifies a key pair generator field used for form in HTML5 . This element is not yet been supported by Internet explorer. So it is recommended not use this element.

<output> element: HTML5 introduces a new element called output , which is used to represent different types of outputs like a output return by a script Example:

```
<form action "#" output=" x.value=parseInt(a.value)+parseInt(b value)">
<input type= "number id="a" name="a" value="0">
<output name ="x" for ="a b"> <output>
</form>
```

Here for attribute in output element is used to specify the relationship between the output element and other element in the document that affect the calculation. The value of the "for" attribute is a space – separated list of IDs from other elements.

<progress> and <meter> element: <progress> and <meter> element is used to represent the bar which shows the completion of a task like downloading a file.

Example:

```
<progress value=10 max=100>100</progress>
<meter value=10 max=100></meter>
```

List of attributes used in <progress> and <meter> element is shown in the following table:

Attribute	Description
Value	The point up to which the progress is completed
Min	The lowest value of the progress bar
Low	The point that marks the upper boundary of the low segment
High	The point that marks the lower boundary of the high segment
Max	The highest value of the progress bar
Optimum	The point which is the optimum position for the progress bar

<input> type in HTML5

HTML5 introduces various new input types i.e. you may put various new values for “type” attribute in the input element.

The following table shows new <input> types:

Input Type	Description	Example
color	The input is color in rgb format.	<input type="color" name="color">
date	The input is date.	<input type="date" name="date">
date time	The input is date and time.	<input type="datetime" name="datetime">
datetime-local	The input is local date and time.	<input type="datetime-local" name="local">
email	The input is email.	<input type="email" name="email">

month	The input is month	<input type="month" name="month">
number	The input is number	<input type="number" name="number">
range	The input is range. Default is from 0 to 100.	<input type="range" name="range">
time	The input is time.	<input type="time" name="time">
url	The input is url.	<input type="url" name="url">
week	The input is week.	<input type="week" name="week">

HTML5 introduces new input attributes. The list of new attributes are:

autocomplete, autofocus, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height and width, list, min and max, multiple, pattern (regexp), placeholder, required and step.

There are different ways by which you can use the attributes in HTML5 form tag. Some of the types are:

Type	**Example**
Empty	<input type="text" value="VMOU" disabled>
Unquoted	<input type="text" value=VMOU>
Double-quoted	<input type="text" value=" VMOU">
Single-quoted	<input type="text" value=' VMOU' >

In HTML5, all four syntaxes may be used, depending on what is needed for the attribute. Now we discussed some of the form attributes that are generally used by modern web.

placeholder attribute: If you want to provide some hint in the <input> & <textarea> to the user that what could be the format of input text entered into in. This can be done by the placeholder attribute in the following way:

Example:

```
<input type ="text"name="search" placeholder="search to the web">
<input type ="email" name=" email" placeholder="username@domain
name.com">
```

autofocus attribute: If we use autofocus attribute then the cursor will focus on the particular input element after page loads. This is previously done using JavaScript.

Example:

```
<input type ="text" name="a" autofocus />
```

required attribute: Previously we need client-side validation for the empty input field , now HTML5 introduces a new attribute for validating empty field to be submitted i.e. required attribute.

Example:

```
<input type ="text" name=" uname" required>
```

This will not be submitted until it is filled.

8.5 HTML5 Multimedia

HTML5 also supports multimedia like sound, music, images, animations, videos etc. HTML5 has an element for displaying video, which was previously played with the help of plugin (like flash) in a browser.

HTML5 Video: HTML5 has a standard way to embed a video in a web page via <video>element.

Example:

```
<video width="100" height="100" controls>
<source src="mymovie.mp4" type ="video/mp4>
<source src="mymovie.ogg" type ="video/ogg>
    oops !!video tag is not supported by your browser
   </video>
```

Here,

control attributes adds the control like play, pause and volume.

source tag defines the video clip which you want to play.

If the video element is not supported by your browser then the text between them will be displayed.

It is recommended that the video file should be in mp4 format, because it is supported by most of the browsers.

HTML5 Audio: We can also display audio file without any plugin (like flash). This is done with the help of <audio>tag

Example:

```
<audio width="100" height="100" controls>
<source =src="myaudio.mp3" type ="audio/mp3>
oops !! your browser does not support audio tag.
</audio>
```

HTML5 plugins: Plugins are the helping application which extends the basic functionality of a web browser like JavaScript etc.

For specifying the plugin in HTML5, HTML5 uses <object>tag or <embed>tag.

Example:

```
<object width="400" height="50" data="QCA.pdf">
</object>
```

Here data attribute represents the file you want to show to the web page.

You can also play YouTube video through your own web page using <iframe>tag as:

```
<iframe width="100" height="100"
src="https://www.youtube.com/embed/Io9G7wy09uA">
</iframe>
```

8.6 HTML5 Links

While adding navigation on the web page, we generally add a link. These links are supposed to be one of the most important HTML elements. They represent the connection between the two HTML documents.

The hyperlink in HTML is represented by <a>elements. The syntax is like:

```
<a href="www.vmou.ac.in"> open vmou </a>
```

In above example, we define a hyperlink to the vmou home page using absolute path and the anchor text.

We can also use <a>element to link up content within the same webpage by using Id on the element for that you have to put #id with prefix # in href attribute of <a> element.

Example:

```
<h1 id="header">Hello world</h1>
<p>
Hi again !!!.
</p>
<h2> Goodbye</h2>
<p>
Bye again!!!
<p>
<a href="#header"> move to the top</a>
```

There are many attributes are available in the <a> element these are described in the following table:

Link Attributes	**Description**	**Example**

href	The href should contain a valid URL as
	● absolute address
	● relative address
	● IP with #prefix
		href="www.vmou.ac.in
		href="about.html
		href="#hello"

title	It provides the description of the link. When you put mouse pointer on the link it will display the content of the title.	<a href="vmou.ac.in title="click to enter into vmou homepage">
rel	rel is used to define the relationship of the current page to the linked page.	rel="help"
target	It will specify how the link page is open i.e. either in new tab/window or in the current window.	**For new window/tab** <a href="vmou.ac.in" target="_blank">vmou</a> **For same window/tab** <a href="vmou.ac.in" target="_self">vmou</a>

		By default some browsers
download	It specifies the browser that the link is downloads a file not to be shown on the web browser.	<a href="script.js" download> click to download</a >
ping	This is used by visitor/tracker and analytics. The link should be opened when user open that link. The link should be notified when the user open that link	<a href="vmou.ac.in" ping="vmou.ac.in/click"> </a>

Link type: There are many links are available in HTML5, which specify the relationship between the current pages that you are linking to. These links can be discussed with the help of table:-

Link type	**Description**
alternate	Links to the alternate representation (may be a copy of same web page in different language) of web page.
author	Link to the author page of the document or article
bookmark	The link specifies to bookmark the web page /document
help	Move to the web page which contains help of the current article.
license	Specifies the link page contain the copyright or licensing.
next	Links to the next page from the series of pages.
nofollow	Indicates that the current document's original author or publisher does not endorse the referenced document. This attribute is often

	used to declare paid links to search engines such as Google, who, request that webmasters declare all paid links (e.g., advertising) in this manner.
prefetch	This specifies the browser to fetch the linked page in advance so that when user clicked on that link, He or she can get the page quickly.
prev	Links to the previous page in the series of the pages.
search	Link to resource that can be used to search through the current page or related pages.
tag	This is generally used in the blog

8.7 API of HTML5

HTML Geolocation

HTML5 introduced a new API (Application program Interface) which is used to get the geographic location of the devices (usually GPS enabled) called HTML5 Geolocation API.

Before using this API please ensure the following things:

- Since, geolocation API comprises with the user privacy so it need the approval of user to get the location.
- The browser should support geolocation API.

In this section we use getCurrentPosition () method, which is returns an object 'position'. The 'position' object contains number of properties like latitude and longitude of the user position. Further, we use these properties to calculate the distance between the user and some object.

Follow the following steps to get user Geolocation:

1. First you need to detect user browser support the Geolocation API, for that you can use the following code :-

```
If (navigator.geolocation)
{
  alert('Geolocation supported');
}
```

'navigator' is an object which contains information about the browser like version number, browser vender etc. The browser which support Geolocation API has a geolocation property. It is good to check geolocation API is supported by your browser with the help of if statement as above.

2. If browser has Geolocation API then you can request his/her device location with the help of getCurrentPosition() function as:

```
navigator.geolocation.getCurrentPosition (function (position)
{
        //code to use Geolocation.
});
   --------
```

getCurrentPosition() function take another function as an argument and in that function you create a variable with name 'position ' variable is used to obtain user's Longitude and Latitude.

3. To get the longitude and latitude coordinate, you need to select 'coords' property as

   ```
   position.coords.latitude;
   position.coords.latitude;
   ```

Now we use these steps to get the location of user .The following example pop up an user's latitude and longitude as an alert message when user click on the 'click me' button.

Example:

```
<html>
     <body>
     <p> click on the button and you will get your longitude and latitude
</p>
    <button onclick="getlocation()">click Me</button >
```

```
      <script>
function getlocation()
{
   if (navigator.geolocation)
    {
      navigator.geolocation.getcurrentPosition(
      function (position){
      alert ("latitude:"+position.coords.latitude +
"longitude:"+position.coords.longitude;
    });
}
else
{
alert ("Geolocation is not supported by your browser");
}
}
</script>
</body>
</html>
```

There are various properties other than longitude and latitude available in HTML5 that can be used with getcurrentPosition() method. They are:-

Properties	**Description**
coords.latitude	returns the latitude as a decimal number
coords.longitude	returns the longitude as a decimal number
coords.accuracy	returns the accuracy of position
coords.altitude	returns the altitude in meter above the sea level
coords.altitudeAccuracy	returns the altitude accuracy of position
coords.heading	returns degrees clockwise from north
coords.speed	returns speed in meter/sec.

timestamp	returns date /time of the response

The other methods for geolocation objects are:-

Method	Description
watchposition()	This method continuously returns the current position of the user. This is used when user is continuously moving.
clearwatch()	This will cancel the ongoing watch position()

The above properties and method can be used in sophisticated web application like

- Turn by turn navigation.
- Shows points of interest near you. (This task can be done using a function called haversine(latitude1,longitude1, latitude2, longitude2,), which is used to calculate distance between two object by taking its corresponding longitude and latitude).

HTML5 Drag and Drop

Drag and Drop is a process in which user grab an object and drag it to a different position. HTML5 has introduced the new API which support drag and drop mechanism.

For better understanding drag and drop API follow the following steps:-

1. First you should make the element draggable by setting draggable attribute to true. Let us say we want an image to be dragged then we should write as

```
<img src="1.jpg" draggable="true">
```

2. Then we specify what should happens when user drag the element/data. This is done using a function which will be called on the event on drag start as
   <img src="1.jpg" draggable =true" ondragstart="drag(event)">
   When we drag the image a function drag (event) is called.
   The definition of drag(event)function will be set the data type and the value of the dragged data.

```
function drag(event)
  {
    event.dataTransfer.setData("text",event.target.id);
  }
```

The dataTransfer.setData()method set the data type and value of dragged data. In the above example data type is text and value is id of draggable element.

3. When the drag is over then we must present the default handling of the data/element cannot be dropped in other element. This can be done using calling a function ondragover event as
   <div id ="drop1" ondragover= "allowdrag (event)" > </div>
   The definition of allowdrag(event) will present in the default handling of the element

```
function allowdrag (event)
{
        event. preventDefault();
}
```

4. Finally we define a function which is called when element/data is dropped. This function will be called on the "ondrop" event.

```
<div id= "drop1" ondragover= "allowdrag(event)" ondrop="drop(event)"> </div>
```

The definition will drop the image to div whose id is, drop1, as a

```
function drop(event)
{
            drop1.innerHTML='<img src="'+ event.dataTransfer.getData('Text') +
'" style="height:100px; width:100px"/>';
}
```

The complete code shown below:

```
<!DOCTYPE HTML>
<html>
<head>
```

```
<style>
#drop1 {
  height: 100px;
  border: 5px solid #ccc;
}
</style>

<script>
function allowDrop(event) {
  event.preventDefault();
}

function drag(event) {
  event.dataTransfer.setData("Text", event.target.id);
}

function drop(event) {
  event.preventDefault();
  drop1.innerHTML='<img src="'+ event.dataTransfer.getData('Text') + '"
style="height:100px; width:100px"/>';
  event.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<p>Drag Example</p>
<p>Drag images to the empty rectangle</p>
```

```
<div id="drop1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<br>
<img src="1.jpg" draggable="true" ondragstart="drag(event)" width="100"
height="100">


<img src="2.jpg" draggable="true" ondragstart="drag(event)" width="100"
height="100">
</body>
</html>
```

HTML5 Client Side Web Storage

In the current scenario, every web site uses some kind of storage for storing data like data about user, page visits, video or audio etc. These stored data can be used by companies as a decision making tool like knowing a user and their behaviour etc.

Traditionally, for storing data many uses server side storage. However, in this section we will discuss alternative called client side storage, in which data stores in the device of client, with the use of client side storage we can:-
(i) Increase the performance of website.

(ii) Provide offline access

(iii) Reduce the dependency upon cookies which has limited stage.

HTML5 has introduced new API for storing data onto Client like session and cookies. They are

(a) Local storage

(b) Session storage

(a) Local storage: - Local storage API is storing megabytes of data link the entire user's mailbox. Local storage has a simple way of storing data using key/data pairs. This API has an object called load storage which provide some functions to add data or access the data. We can access this API using JavaScript. It is further noted that the data stored using local storage Is separate for each web application and not accessed by the other. For example suppose you have to sub domain like a.vmou.ac.in and b.vmou.ac.in then both sub domain will have separate local storage and they cannot be accessed from other domain. The word "sandboxing "refers to assigning data store to specific domain. The following are the detail of functions associated with the local storage object:-

Function name	**Description**	**Example**
setItem(Key, Value)	This function stores the value associated with the key	localStorage.setItem("university","vmou");
getItem(key)	This function is used to return the value associated with the key.	localStorage.getItem("university");
removeItem(key)	This will detect the value associated with key	localStorage.removeItem("university");
key(index)	If you have multiple key values pair them you can also retrieve value using function	for(i=0;i<localStorage.length; i++) { console.log(localStorage.key(i)) ; }
clear()	This will clear all the data store in local storage datastore	localStorage.clear();

There are some properties associated with local storage. These are:-

Property name	**Description**	**Example**
hits	Counts the number of times user hits some	localstorage.hits;

	object	
length	Counts how many key/value pairs stored in localstorage object.	localstorage.length;

Example- In this example we store a key value pair and display its value onto the alert box by

```
<html>
<body>
        <script>
                if(typeof(storage)==”undefined”)
                        {
alert (“oops!!! Your browser does not support web storage”);
                        }
                else
{
localStorage.setitem(“university”,”vmou”)
                        alert(localStorage:getitem(“university”));
}
</script>
</body>
</html>
```

(b) Session storage: - this storage is similar to local storage except it only retains for a session, after that all the data associated with it will be vanish. This API has some functions as local storage API and you can use it in the similar fashion like:-

```
sessionStorage.setItem  (“university”,”vmou”);
sessionStorage.getItem  (“university”);
sessionStorage.key  (0);
sessionStorage.removeItem  (“university”);
sessionStorage.clear ("university")
```

Beside the advantages client side storage it also has some disadvantages like storage limits at client side, cross directory attacks , loss of sensitive data of user etc.

HTML Application Cache

HTML5 introduces application cache by which we can cache the data of web application onto Client device and make the web application offline. By creating cache we can use application offline, the cache resource loads faster hence increase the speed and this will reduce the sever load.

We can cache the resource of web application by creating HTML5 cache manifest file. Manifest file tells the browser what resources are accessed offline and if some assets are missing then should be retrieved from web.

To enable application cache we have to include manifest attribute in <html>tag as:-

```
<! DOC TYPE HTML>

<html manifest= "test.appcache">

<!--write some code -->

</html>
```

In the above html code we include manifest file name "test.appcache". It is recommended that we should save manifest file with ".appcache" file extension.

The manifest file is a simple text file, which generally has three sections:-

CACHE MANIFEST:- This is the first line of manifest file within that there are some file names also known as namespace. These listed namespaces are cached by the browser for offline access.

Example:

```
CACHE MANIFEST

/demo.css

/pic.jpg

/demo.js
```

```
/try.js
```

NETWORK:- This section contains files which will be served In case of an internet connection cannot be establish in place of all files that are offline.

Example:

```
NETWORK:
login.php
```

We can put * in place of file name, which means all the other resources/files are not cached or requires internet.

FALLBACK: This section contains files which will be served in case of an internet cannot established in place of all files that are online.

Example:

```
FALLBACK:
/html/../offline.html
```

The complete MANIFEST file look likes:

```
CACHE MANIFEST
/demo.css
/pic.jpg
/demo.js
/try.js
NETWORK:
Login.php
FALLBACK:
/html/--/offline.html
```

The cache should be updated when:

- The user clears the browser's cache
- The manifest file is modified

This is done by using #, which is used to put comment, on the manifest file. For updating cache resource we put "#"followed by date and time and version of the resource as

#2017-02-17 v/.1.1/

The application cache will updated when manifest file changes.

Example:

```
CACHE MANIFEST
        #2017-02-17   v/.1.1/
        /demo.css
       /pic.jpg
      /demo.js
      /try.js
```

There are some precautions while using application:

- Ensure that the cache resource should be updated and only the updated version is to be shown to the user and for that we need to update the manifest file.
- Many browsers has a limit to store the resources to be cache (generally maximum size is 5MB). So we need to limit the cache resource.

HTML Web Works

HTML5 has a new feature by which we can JavaScript run in background without affecting the performance of the user interface.

A Web Workers is a JavaScript that runs in the background, independently of other scripts, with little affecting the performance of the page. While running a JavaScript you may continue whatever you want to do like clicking, selecting etc. These contain the code that consumes relatively more CPU cycle. Sometimes Web Workers make the system less responsive, so it is not suggested to use extensively and in large numbers.

We follow the simple steps to create and work with Web Workers:

Step-1:- First we have to test whether our web browser supports Web Workers or not. This can be done using following code: -

```
          if (typeof (Worker) != ”undefined”)
            {
              // web worker is supported write some code
            }
           else
           {
             // oops!!! your browser is not supports web worker.
           }
```

Step 2:- If the web worker is supported then we are ready to create a new worker:-

```
var worker =new worker (“heavy_task.js”)
```

The above worker is initialized with the URL, which contains code that will executed by worker. If the URL is valid and found a JavaScript file then browser will create a new worker thread which downloaded asynchronously. An error of 404 will be shown if the file at the specified location is not found.

Step3:- When new worker object is running and ready to be used then we have to use postMessage method for communication between web worker and present page.

The postMessage() method accepts a string or JSON object as a single argument. The use of postMessage is shown in heavy.task.js

```
var j, count=0;
for (var i=0;i<10000 ;i++)
 {
   for (j=0; j<10000;j++)
    }
    count++;
    }
 }
postMessage(count);
```

Step 4:- Now we will receive the value of count from the worker via onmessage event handler as

```
worker.onmessage=function (event){
alert ('worker count value==' + event.data);
}
```

The coding of main page look like:

```
<html>
<head>
<title> Test webworker<\title>
</head>
<script>
var worker  =new Worker("heavy_task.js");
worker.onmessage = function(event){
alert ('worker count value=='+event.data);
}
</script>
</html>
```

Step 5:- Web Workers will not stop automatically we have to stop by calling terminate method as follows:-

```
worker.terminate();
```

After terminating the worker the task related to the worker will not respond. It is noted that we cannot restart the same worker instead we can create a new worker using the URL by following these simple steps also we can create many sophisticated applications like Messaging API etc.

It is noted that by the worker we are not able to access the following JavaScript object:-

- The window object
- The document object
- The parent object

This is because the web workers are in external file.

HTML Real Time

HTML5 supports real time services in which either server or client can communicate and achieve real time communication. These can be used in application like updating the price of product on a website, News feeds, receiving updates from the server, transferring data bidirectional etc.

There are two ways to add real time services to your web application:-

(i) Web sockets

(ii) Server-send event

(i) HTML5 Web sockets: This API allows creating a TCP connection between server and client which allows two ways, real time communication between them. Websocket operator over a single socket and exposed via JavaScript interface in HTML5 compliant browser. The Websocket is very easy to use, as we had already discussed the working of postMessage () method and on message event in web worker API.The Websocket API also works in the similar fashion.

First we have to create a new Socket as:-

```
var socket =new websocket (url, [protocol]);
```

Example:

```
var socket =new websocket ("ws: //localhost:8080/");
```

It is noted that the protocol of URL must be ws: // and the rest of the URL can be structured as it is.The protocol is optional and specifies the protocol used for connection to be successful.

Now we will discuss the attributes with the Websocket object. The list of attributes and its description are shown in the table:

Attributes	Description
Socket.readystate	This is a read only attribute and return the values in

	between 0-3. Each values represent the state of connection as: If value is 0 then connection has not been established If value is 1 then connection is established and communication is possible If value is 2 then connection is going through the closing handshake. If value is 3 then connection has been closed and could not be opened
Socket.bufferedAmount	This is also read only attribute which returns the value of number of bytes of text that has been queued using send method.

There are also some events associated with web sockets object. They are shown in the table:

Event	Description
Socket.onopen	This event occurs when socket connection is opened
Socket.onmessage	This event occurs when client receives a data /message from server
Socket.onerror	This event triggers when there is an error in communication
Socket.onclose	This event triggers when connection is closed.

Following table shows the methods associated with websocket real time service along with their description:-

Method	Description
Socket.send	This method is using connection to send data
Socket.close	This method will terminate the established connection.

Websocket Example:

```
if (“websocket” in window)
{
         var socket=new WebSocket(“ws://localhost:8080/chat”);
         socket.onclose=function (event){
         alert(‘connection is closed’);
         }
socket.onmessage=function (event){
         alert(‘received message is’+event.data);
         }
socket.onopen=function (event){
         alert(‘socket is opened now you can send or receive message’);
             Socket.send(‘Hello server’);
             }
         }
         else
              {
alert (‘web socket is not supported’);
}
```

The above example shows how to send and receive a message using HTML5 websocket API.

For testing we need a server program which also supports websocket API for communication between client and server.

For sending the update to the client, we need server capabilities to sending updates (like in PHP and ASP) which we can write with php script as (demo-sse.php)

Example:

```
         <?php
```

```
        header('Content-Type:text/event-stream');
        header ('Cache-Control:no-cache');
        echo "{data:server time is {data('r');}"
        flush() ;
         ?>
```

First we have to set the "Content Type" header to "text/event" streams and Cache-Control to no-cache and then we can start sending event streams by starting the strings with "data:" and finally flush the output data back to the web page using flush() method.

Event source object has onopen , onmessage and onerror events which are similar to webSocket object.

(ii) HTML Server Send Events: As websocket provides a bi-directional communication between server and client but sometimes we need one way information to the server like simple push message, getting updates from server in that case we need HTML5 server send events. HTML5 server send event (SSE) is an API which flows from web server to the web browser. Using SSE you can push DOM events continuously from your web server to the client browsers

Server Send Event comes with event source object. They are quite similar to web socket while using it we can receive Server Send Event notification as

```
var sse = new EventSource ("server-sse.php");
sse.onmessage= function (event) {
document.getElementbyId ("demo").innerHTML=  event.data+"<br>"; }
```

In the above code we create an object of Event Source and specify the page in URL from which we received the update (server-sse.php)

The complete code at client side looks like

```
<html>
<body>
  <h1> Testing SSE<1/h1>
```

```
 <div id= “demo”><1div>
<script>
      If (type of (Event Source)!= “undefined”){
        var sse=new Event Source (“server-sse.php”);
   sse.onmessage= function (event) {
document.getElement by Id (“demo”).inner HTML+= event.data+”<br>”;
                       };
           }
   else
      {
        alert(“ oops !!! your browser does not support SSE”);
          }
</script>
</body>
</html>
```

For sending the update to the client, we need server capabilities to sending updates (like in Php and ASP) which we can write with php script as

```
Example(demo-sse.php)-
        <?php
         header(‘content-type:text/event-stream’);
         header (‘cache-control:no-cache’);
         echo “{data:server time is {data(‘r’);}”
         flush() ;
          ?>
```

8.8 Self Learning Exercise

Q.1 Which of the following is not a sementic element in HTML5?

a) <article>

b) <aside>

c) <mark>

d) <table>

Q. 2 Which of the following element is use to show complition of a task in HTML5?

a) <progress> and <meter>.

b) <keygen>

c) <output>

d) <datalist>

Q.3 Which of the following API is used for storing data onto Client like session and cookies in HTML5?

a) HTML5 Geolocation API.

b) HTML5 server send event (SSE).

c) HTML5 Audio API.

d) Local Storage and Session Storage.

8.9 Summary

In this chapter you learned about HTML5 and summarized as:

- HTML5 introduces these semantic elements for better structure and easy for search engine to identify the content of web page.
- HTML5 introduces new input attributes like autocomplete, autofocus, form, formaction, formenctype.
- There are some new semantics elements introduced in HTML5 like <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header> etc.
- <acronym>, <applet>, <basefont>, <big>, <center>, <dir>, <font>, <frame>, <frameset>, <noframes>. <strike>, and <tt> tags that are deleted from HTML5.

- HTML5 also introduce many new APIs/features like Geolocation, Drag and Drop, Local Storage etc.
- HTML5 has an improvement to its HTML classical form by introducing various new attributes, elements and input types. like <datalist>, <keygen>, <output> etc.
- We can also display audio file without any plugin (like flash) with the help of <audio>tag.
- HTML Geolocation is used to get the geographic location of the devices (usually GPS enabled) called HTML5 Geolocation API.
- HTML5 has introduced the new API which support drag and drop mechanism. This can be done by the draggable="true" property.
- HTML5 has an API for storing data onto Client like session and cookies they are Local Storage and Session Storage.
- HTML5 also has a feature by which we can JavaScript run in background without affecting the performance of the user interface.
- HTML5 Web sockets is used for creating a TCP connection between server and client which allows two ways, real time communication between them.
- HTML5 server send event (SSE) is an API which flows from web server to the web browser.

8.10 Glossary

Semantic- The word 'semantics' means study of the meaning of words and phrases in a paragraph.

API- Aplication program Interface.

8.11 Answers to Self-Learning Exercise

Q.1 d

Q.2 a

Q.3 d

8.12 Exercise

Q.1 Explain HTML5 Client Side Storage in detail.

Q.2 What is geolocation API in HTML5? Explain with an example.

References and Suggested Readings

1. Matt West, HTML5 Foundations, John Wiley and Sons, Ltd, Publication, First Edition.
2. Steve Suehring and Janet Valade, PHP, MySQL®, JavaScript & HTML5 All-in-One For Dummies, John Wiley & Sons Publication.
3. www.w3schools.com.
4. www.tutorialpoint.com.

UNIT-9
Cloud Computing and Deployment

Structure of the Unit

9.9 Exercise

9.0 Objective

This chapter provides a general overview of

- Cloud Computing Basics
- Cloud Service
- Cloud Delivery Models
- Cloud Deployment models

9.1 Introduction

The National Institute of Standards and Technology (NIST) characterizes cloud computing as ". . . a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

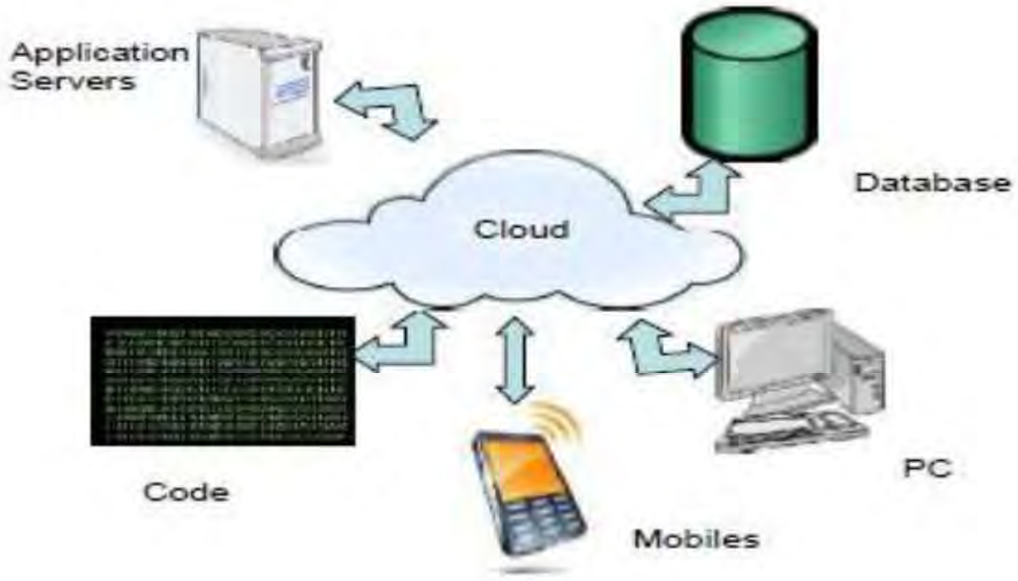


Figure 9.1 A Cloud Computing Environment

In others words it can be defined as:

"Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources."

While there are countless other definitions, there seems to be common characteristics between the most notable ones listed above, which a cloud should have: (i) pay-per-use (no ongoing commitment, utility prices); (ii) elastic capacity and the illusion of infinite resources; (iii) self-service interface; and (iv) resources that are abstracted or virtualised.

9.2 Basic Characteristics

In this section we describe the essential characteristics that a cloud must possess. Any cloud is expected to have these five characteristics that are being described below.

A. On-demand self-service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

B. Broad Network Access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants (PDAs)).

C. Resource Pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the subscriber generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data centre). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

D. Rapid Elasticity

Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

E. Measured Service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

9.3 Basics of Virtualization

Virtualization is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources. It is the process by which one computer hosts the appearance of many computers. Virtualization is used to improve IT throughput and costs by using physical resources as a pool from which virtual resources can be allocated.

Virtualization is a fundamental part of cloud computing, especially in delivering Infrastructure as a Service (IaaS). Exploring different techniques and architectures of the virtualization helps us understand the basic knowledge of virtualization and the server consolidation in the cloud with x86 architecture.

9.3.1 Types of virtualization

In cloud computing the virtualization can be done in two ways either by storage virtualization or by software virtualization.

a. Storage virtualization: The storage available is virtualized to get large virtual storage access and it is further used for allocating memory to the cloud clients.

b. Software virtualization: software built by the company can be used by a large number of systems at the same time with the help of virtualization. A virtual layer is created on which the software is installed and used.

9.3.2 Benefits of Virtualization:

With the help of virtualization we can increase the use of resources available to us in many to get more benefits. We should virtualize because of the following reasons:

a. **Isolation among users:** One user should be isolated from the other users so that he/she may not get information about the others user's data and usage and cannot even access other's data.

b. **Resource sharing:** A big resource can be fragmented into multiple virtual resources so that it can be used by multiple users using virtualization technique.

c. **Dynamical resources:** Reallocation of resources such as storage and computational resources is very difficult but if they are virtualised then they can be easily re-allocated.

d. **Aggregation of resources:** The small resources available can be increased at a large extent with the help of virtualization.

9.4 Elasticity, Resiliency, On-Demand Usage, Measured Usage

9.4.1 Elasticity

In cloud computing, **elasticity** is defined as "the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible". Elasticity is a defining characteristic that differentiates cloud computing from previously proposed computing paradigms, such as grid computing. The dynamic adaptation of capacity, e.g., by altering the use of computing resources, to meet a varying workload is called "elastic computing"

Elasticity is an automated ability of a cloud to transparently scale IT resources, as required in response to runtime conditions or as pre-determined by the cloud consumer or cloud provider. Elasticity is often considered a core justification for the adoption of cloud computing, primarily due to the fact that it is closely associated with the Reduced Investment and Proportional Costs benefit. Cloud providers with vast IT resources can offer the greatest range of elasticity.

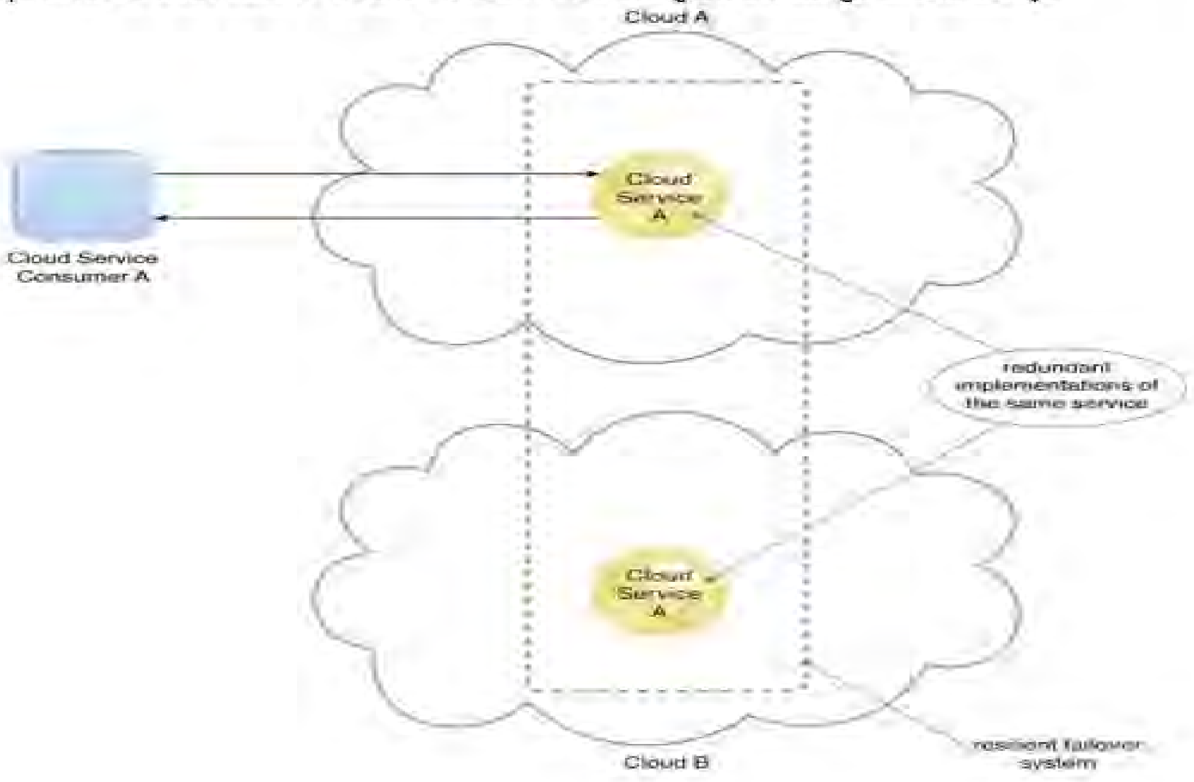

Cloud A
Cloud Service A
Cloud Service Consumer A
redundant implementations of the same service
Cloud Service A
resilient failover system
Cloud B

Figure 9.2 - A resilient system in which Cloud B hosts a redundant implementation of Cloud Service A to provide failover in case Cloud Service A on Cloud A becomes unavailable.

9.4.2 Resiliency

Resilient computing is a form of failover that distributes redundant implementations of IT resources across physical locations. IT resources can be pre-configured so that if one becomes deficient, processing is automatically handed over to another redundant implementation. Within cloud computing, the characteristic of resiliency can refer to redundant IT resources within the same cloud (but in different physical locations) or across multiple clouds. Cloud consumers can increase both the reliability and availability of their applications by leveraging the resiliency of cloud-based IT resources (Figure 9.2)

9.4.3 On-Demand Usage

A cloud consumer can unilaterally access cloud-based IT resources giving the cloud consumer the freedom to self-provision these IT resources. Once configured, usage of the self-provisioned IT resources can be automated, requiring no further human involvement by the cloud consumer or cloud provider. This results in an *on-demand usage* environment. Also known as "on-demand self-service usage," this characteristic enables the service-based and usage-driven features found in mainstream clouds.

9.4.4 Measured Usage

The *measured usage* characteristic represents the ability of a cloud platform to keep track of the usage of its IT resources, primarily by cloud consumers. Based on what is measured, the cloud provider can charge a cloud consumer only for the IT resources actually used and/or for the timeframe during which access to the IT resources was granted. In this context, measured usage is closely related to the on-demand characteristic.

Measured usage is not limited to tracking statistics for billing purposes. It also encompasses the general monitoring of IT resources and related usage reporting (for both cloud provider and cloud consumers). Therefore, measured usage is also relevant to clouds that do not charge for usage (which may be applicable to the private cloud deployment model described in the upcoming *Cloud Deployment Models* section).

9.5 Cloud Resource Administrator

A *cloud resource administrator* is the person or organization responsible for administering a cloud-based IT resource (including cloud services). The cloud resource administrator can be (or belong to) the cloud consumer or cloud provider of the cloud within which the cloud service resides. Alternatively, it can be (or belong to) a third-party organization contracted to administer the cloud-based IT resource.

For example, a cloud service owner can contract a cloud resource administrator to administer a cloud service (Figures 9.3 and 9.4)

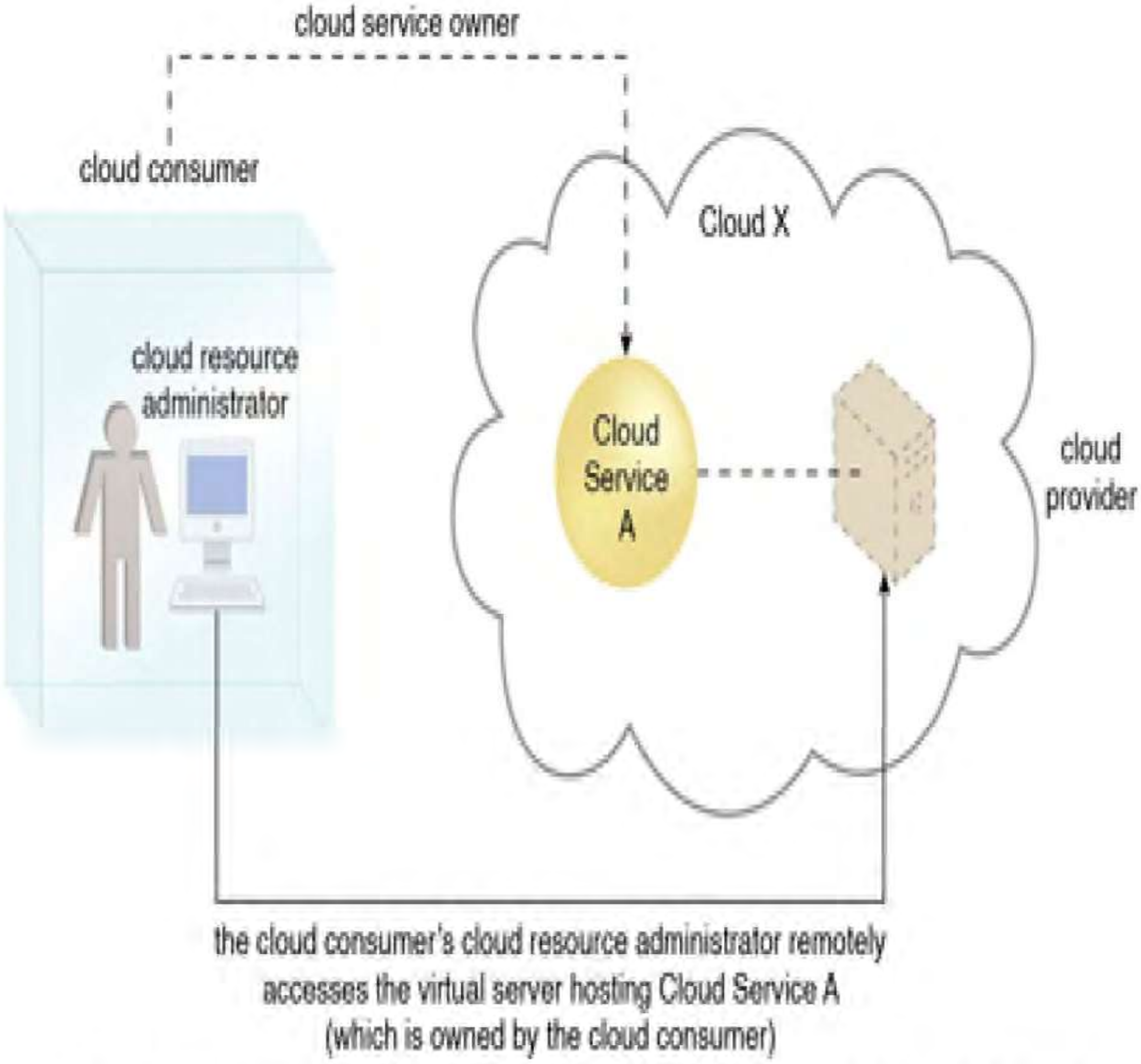

cloud service owner
cloud consumer
Cloud X
cloud resource administrator
Cloud Service A
cloud provider
the cloud consumer's cloud resource administrator remotely accesses the virtual server hosting Cloud Service A (which is owned by the cloud consumer)

Figure 9.3- A cloud resource administrator can be with a cloud consumer organization and administer remotely accessible IT resources that belong to the cloud consumer.

The reason a cloud resource administrator is not referred to as a "cloud service administrator" is because this role may be responsible for administering cloud-based IT resources that don't exist as cloud services. For example, if the cloud resource administrator belongs to (or is contracted by) the cloud provider, IT resources not made remotely accessible may be administered by this role (and these types of IT resources are not classified as cloud services).

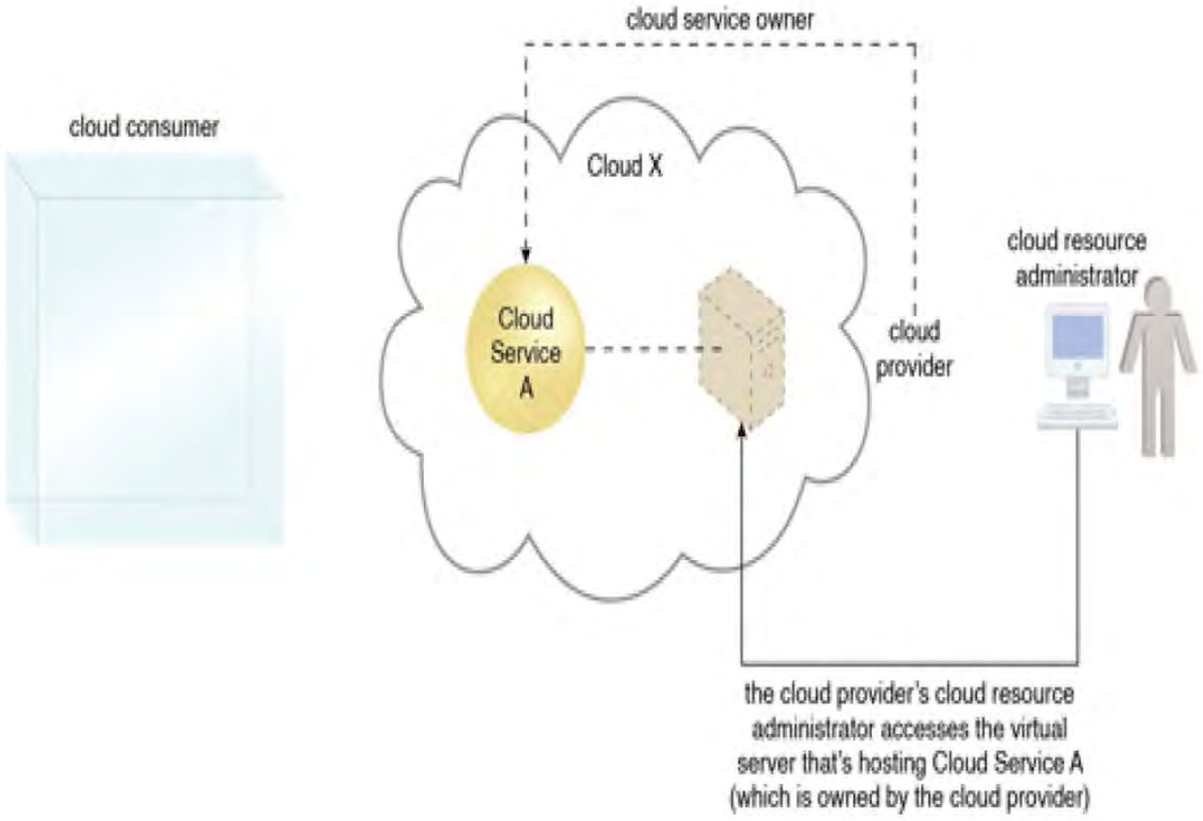

cloud service owner
cloud consumer
Cloud X
Cloud Service A
cloud provider
cloud resource administrator
the cloud provider's cloud resource administrator accesses the virtual server that's hosting Cloud Service A (which is owned by the cloud provider)

Figure 9.4 - A cloud resource administrator can be with a cloud provider organization for which it can administer the cloud provider's internally and externally available IT resources.

9.5.1 Cloud Service Owner

The person or organization that legally owns a cloud service is called a *cloud service owner*. The cloud service owner can be the cloud consumer, or the cloud provider that owns the cloud within which the cloud service resides.

For example, either the cloud consumer of Cloud X or the cloud provider of Cloud X could own Cloud Service A (Figures 9.5 and 9.6).

Note that a cloud consumer that owns a cloud service hosted by a third-party cloud does not necessarily need to be the user (or consumer) of the cloud service. Several cloud consumer organizations develop and deploy cloud services in clouds owned by other parties for the purpose of making the cloud services available to the general public.

The reason a cloud service owner is not called a cloud resource owner is because the cloud service owner role only applies to cloud services.

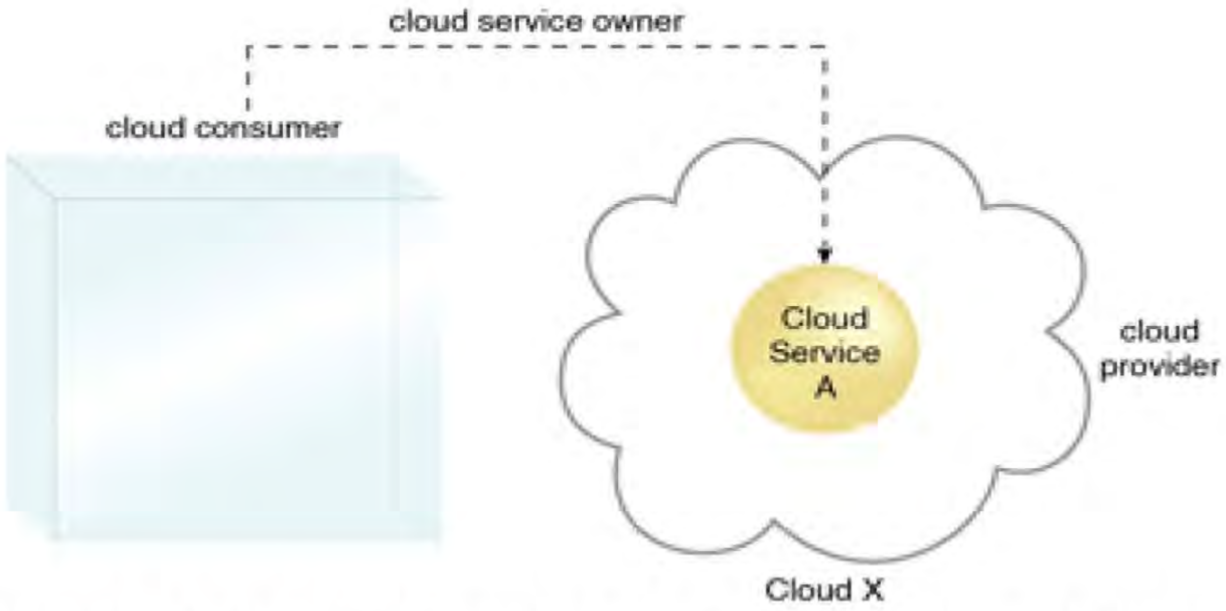


Figure 9.5- A cloud consumer can be a cloud service owner when it deploys its own service in a cloud.

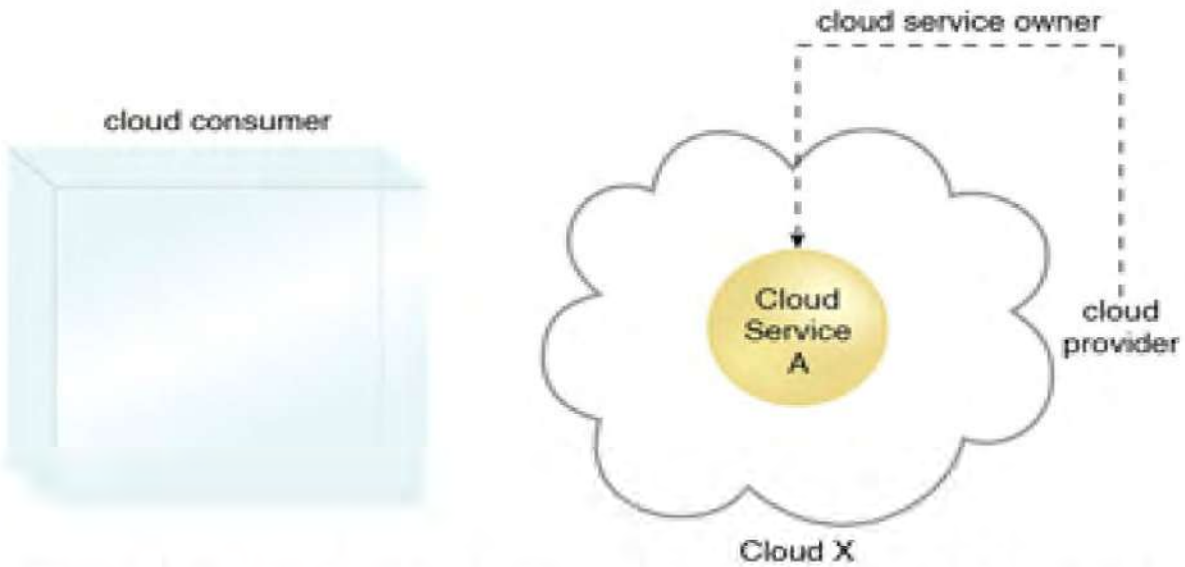


Figure 9.6 - A cloud provider becomes a cloud service owner if it deploys its own cloud service, typically for other cloud consumers to use.

9.5.2 Cloud Service Consumer

The *cloud service consumer* is a temporary runtime role assumed by a software program when it accesses a cloud service.

As shown in Figure 9.7, common types of cloud service consumers can include software programs and services capable of remotely accessing cloud services with published service contracts, as well as workstations, laptops and mobile devices running software capable of remotely accessing other IT resources positioned as cloud services.

software program
service
workstation
laptop
mobile device

Figure 9.7 - Examples of cloud service consumers. Depending on the nature of a given diagram, an artifact labeled as a cloud service consumer may be a software program or a hardware device (in which case it is implied that it is running a software program capable of acting as a cloud service consumer).

9.6 Cloud Delivery Models

A *cloud delivery model* represents a specific, pre-packaged combination of IT resources offered by a cloud provider. Three common cloud delivery models have become widely established and formalized:

- Infrastructure-as-a-Service (IaaS)
- Platform-as-a-Service (PaaS)
- Software-as-a-Service (SaaS)

These three models are interrelated in how the scope of one can encompass that of another.

9.6.1 Infrastructure-as-a-Service (IaaS)

The IaaS delivery model represents a self-contained IT environment comprised of infrastructure-centric IT resources that can be accessed and managed via cloud service-based interfaces and tools. This environment can include hardware, network, connectivity, operating systems, and other "raw" IT resources. In contrast to traditional hosting or outsourcing environments, with IaaS, IT resources are typically virtualized and packaged into bundles that simplify up-front runtime scaling and customization of the infrastructure.

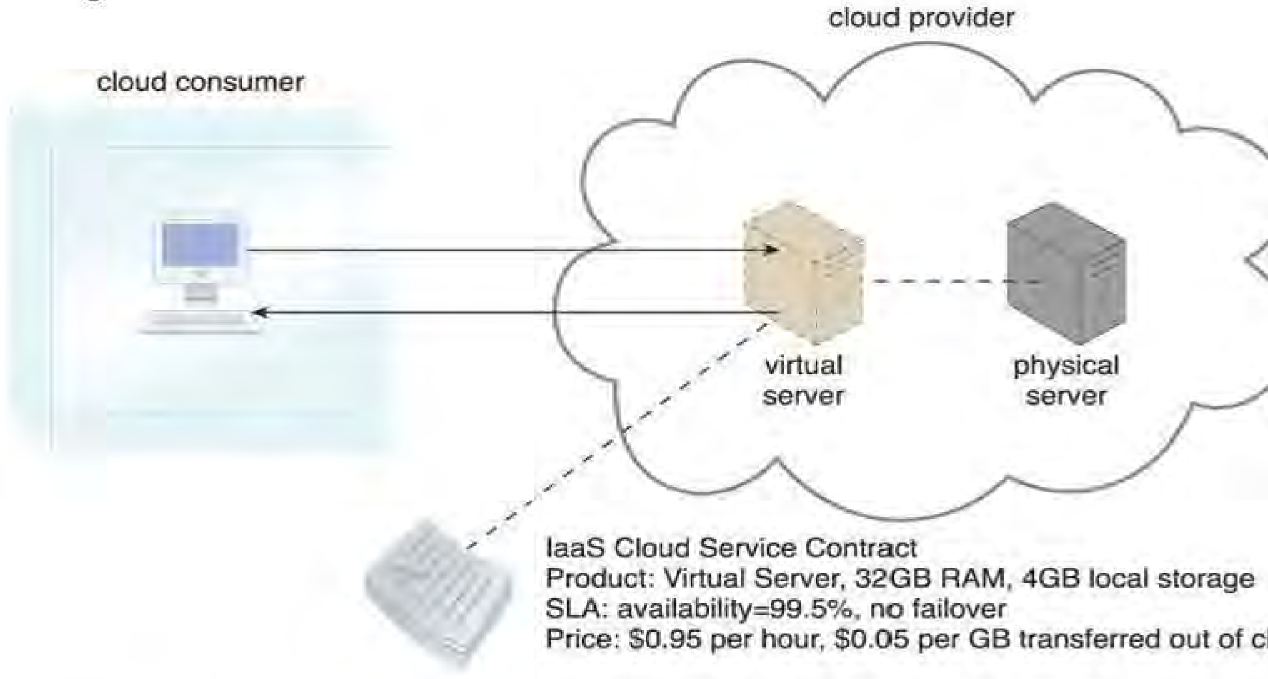

cloud provider
cloud consumer
virtual server
physical server
IaaS Cloud Service Contract
Product: Virtual Server, 32GB RAM, 4GB local storage
SLA: availability=99.5%, no failover
Price: $0.95 per hour, $0.05 per GB transferred out of c

Figure 9.8. A cloud consumer is using a virtual server within an IaaS environment. Cloud consumers are provided with a range of contractual guarantees by the cloud provider, pertaining to characteristics such as capacity, performance, and availability.

The general purpose of an IaaS environment is to provide cloud consumers with a high level of control and responsibility over its configuration and utilization. The IT resources provided by IaaS are generally not pre-configured, placing the administrative responsibility directly upon the cloud consumer. This model is

therefore used by cloud consumers that require a high level of control over the cloud-based environment they intend to create.
Sometimes cloud providers will contract IaaS offerings from other cloud providers in order to scale their own cloud environments. The types and brands of the IT resources provided by IaaS products offered by different cloud providers can vary. IT resources available through IaaS environments are generally offered as freshly initialized virtual instances. A central and primary IT resource within a typical IaaS environment is the virtual server. Virtual servers are leased by specifying server hardware requirements, such as processor capacity, memory, and local storage space, as shown in Figure-8.

9.6.2 Platform-as-a-Service (PaaS)

The PaaS delivery model represents a pre-defined "ready-to-use" environment typically comprised of already deployed and configured IT resources. Specifically, PaaS relies on (and is primarily defined by) the usage of a ready-made environment that establishes a set of pre-packaged products and tools used to support the entire delivery lifecycle of custom applications.

Common reasons a cloud consumer would use and invest in a PaaS environment include:

- The cloud consumer wants to extend on-premise environments into the cloud for scalability and economic purposes.
- The cloud consumer uses the ready-made environment to entirely substitute an on- premise environment.
- The cloud consumer wants to become a cloud provider and deploys its own cloud services to be made available to other external cloud consumers.

By working within a ready-made platform, the cloud consumer is spared the administrative burden of setting up and maintaining the bare infrastructure IT resources provided via the IaaS model. Conversely, the cloud consumer is granted a lower level of control over the underlying IT resources that host and provision the platform (Figure -9).

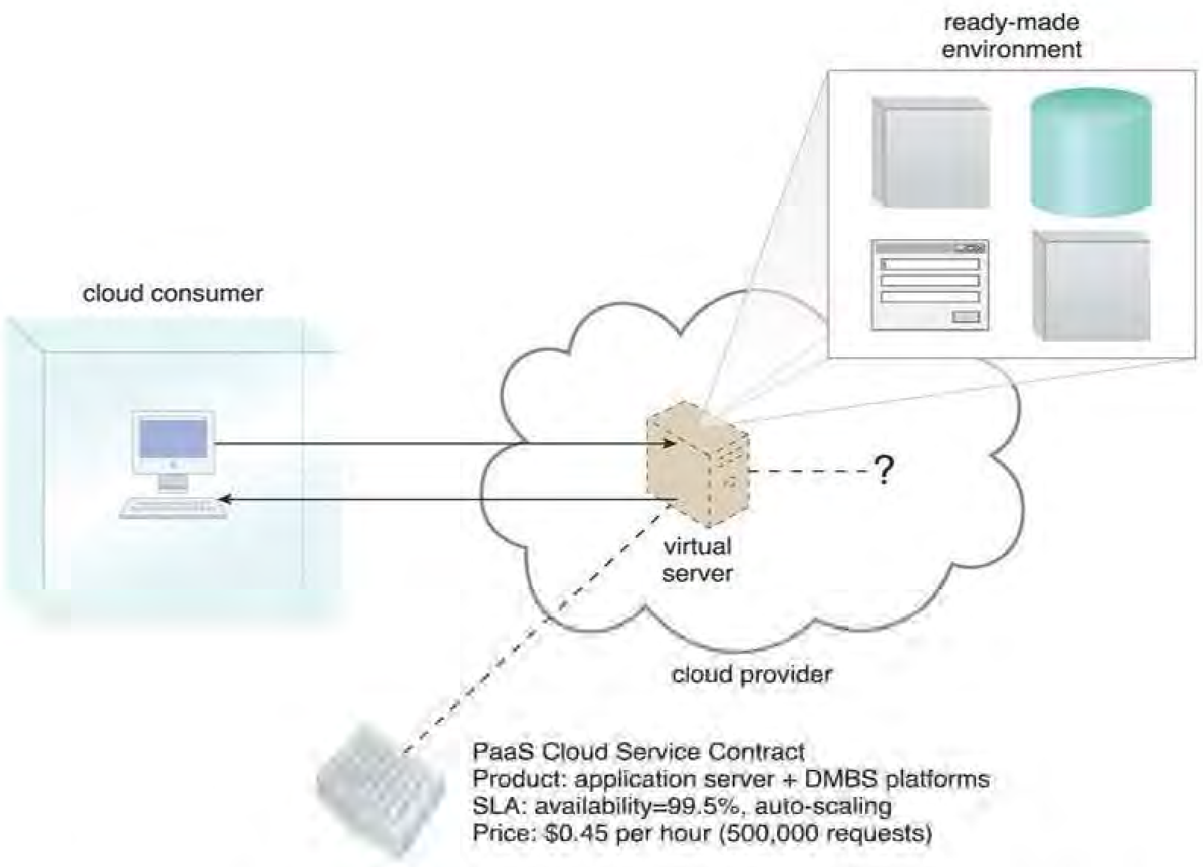

ready-made environment
cloud consumer
?
virtual server
cloud provider
PaaS Cloud Service Contract
Product: application server + DMBS platforms
SLA: availability=99.5%, auto-scaling
Price: $0.45 per hour (500,000 requests)

Figure 9.9. A cloud consumer is accessing a ready-made PaaS environment. The question mark indicates that the cloud consumer is intentionally shielded from the implementation details of the platform.

PaaS products are available with different development stacks. For example, Google App Engine offers a Java and Python-based environment.

9.6.3 Software-as-a-Service (SaaS)

A software program positioned as a shared cloud service and made available as a "product" or generic utility represents the typical profile of a SaaS offering. The SaaS delivery model is typically used to make a reusable cloud service widely available (often commercially) to a range of cloud consumers. An entire marketplace exists around SaaS products that can be leased and used for different purposes and via different terms (Figure-10).

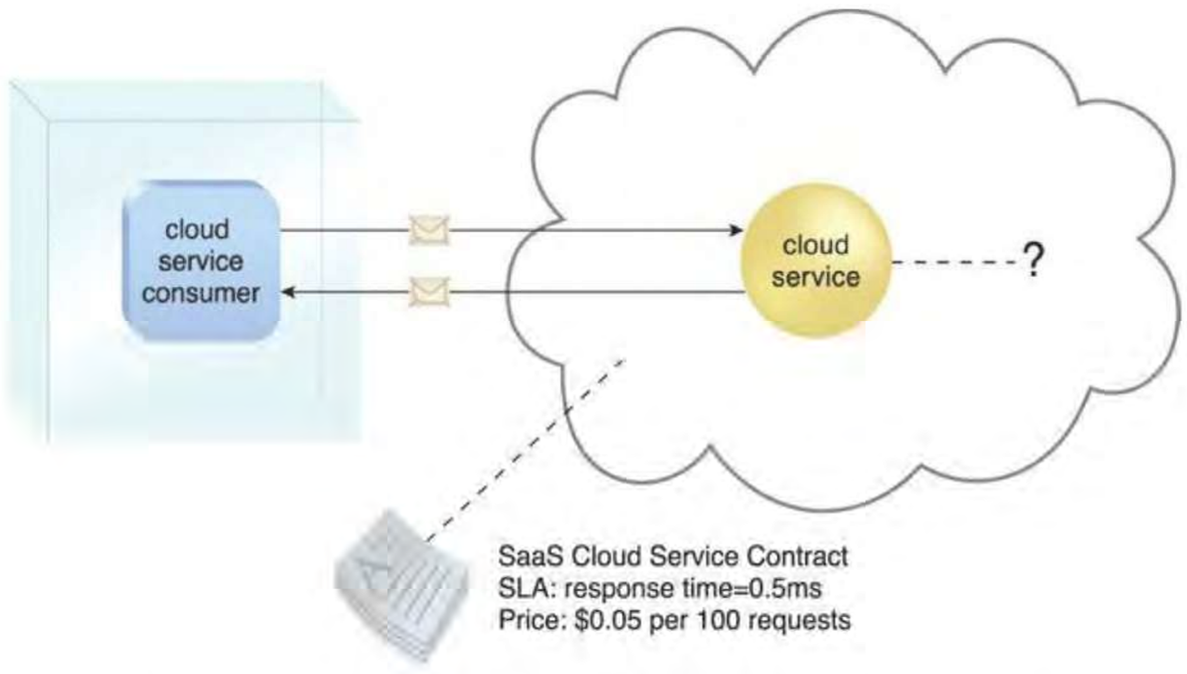


Figure 9.10. The cloud service consumer is given access the cloud service contract, but not to any underlying IT resources or implementation details.

A cloud consumer is generally granted very limited administrative control over a SaaS implementation. It is most often provisioned by the cloud provider, but it can be legally owned by whichever entity assumes the cloud service owner role. For example, an organization acting as a cloud consumer while using and working with a PaaS environment can build a cloud service that it decides to deploy in that same environment as a SaaS offering. The same organization then effectively assumes the cloud provider role as the SaaS-based cloud service is made available to other organizations that act as cloud consumers when using that cloud service.

9.6.4 Comparing Cloud Delivery Models

Provided in this section are three tables that compare different aspects of cloud delivery model usage and implementation. Table-1 contrasts control levels and Table-2 compares typical responsibilities and usage.

Table-9.1 A comparison of typical cloud delivery model control levels.

Cloud Delivery model	Typical Level of Control Granted to Cloud Consumer	Tyoical Functionality Made Acailable to Cloud Consumer
SaaS	usage and usage-related	access to front-end user-

	configuration	interface
PaaS	limited administrative	moderate level of administrative relevant to cloud control IT resources relevant to cloud consumer's to usage of platform
IaaS	full administrative	fullaccess to virtualized infra-structure-related IT rsources and, possibly. To underlying physical IT resources

Table-9.2 Typical activities carried out by cloud consumers and cloud providers in relation to the cloud delivery models.

Cloud Delivery Model	Common Cloud Consumer Activities	Common Cloud Provider Activities
SaaS	uses and configures cloued service	implements, manages, and maintains cloud service monitors usage by cloud consumers
Paas	delops. Tests, deploys, and manages cloud services and cloud-based solutions	pre-configures platfrom and provi-sions underlying infrastructure, middleware. And other needed IT resources, as necessary monitors usages by cloud consumers
IaaS	sets up and configures bare infrastructure, and installs, manages, and monitors any needs	provisions and managers, strorage, networking, and hosting required monitors usages by cloud

	software	consumers

9.7 Cloud Deployment models

A cloud deployment model represents a specific type of cloud environment, primarily distinguished by ownership, size, and access.

There are four common cloud deployment models:

- Public cloud
- Community cloud
- Private cloud
- Hybrid cloud

The following sections describe each.

9.7.1 Public Clouds

A *public cloud* is a publicly accessible cloud environment owned by a third-party cloud provider. The IT resources on public clouds are usually provisioned via the previously described cloud delivery models and are generally offered to cloud consumers at a cost or are commercialized via other avenues (such as advertisement).

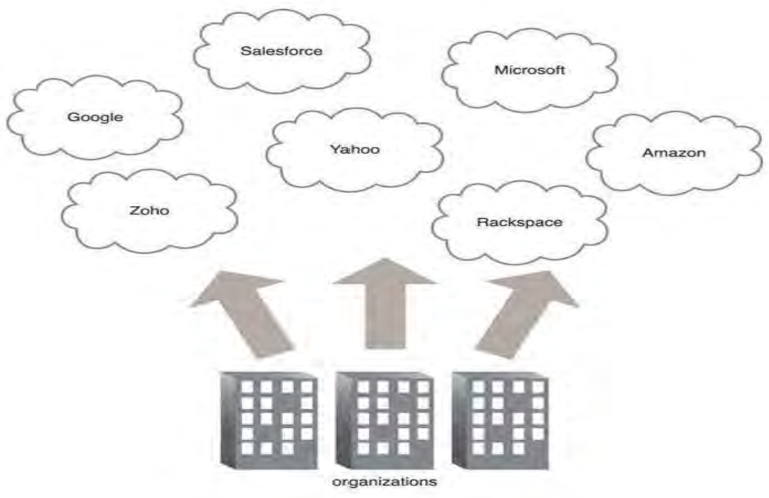


Figure 9.11. Organizations act as cloud consumers when accessing cloud services and IT resources made available by different cloud providers.

The cloud provider is responsible for the creation and on-going maintenance of the public cloud and its IT resources. Many of the scenarios and architectures explored in upcoming chapters involve public clouds and the relationship between the providers and consumers of IT resources via public clouds. Figure-11 shows a partial view of the public cloud landscape, highlighting some of the primary vendors in the marketplace.

9.7.2 Community Clouds

A community cloud is similar to a public cloud except that its access is limited to a specific community of cloud consumers. The community cloud may be jointly owned by the community members or by a third- party cloud provider that provisions a public cloud with limited access. The member cloud consumers of the community typically share the responsibility for defining and evolving the community cloud (Figure)

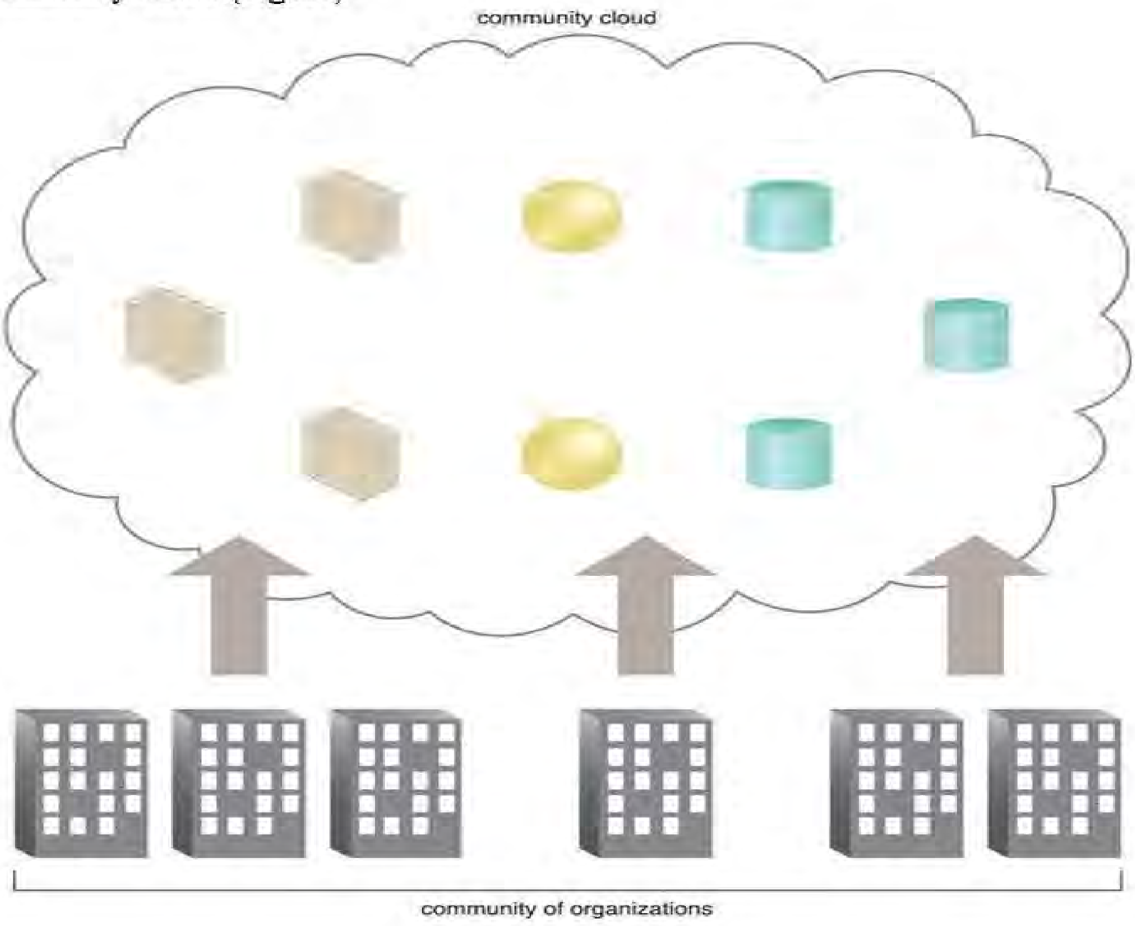


Figure-9.12 . An example of a "community" of organizations accessing IT resources from acommunity cloud.

Membership in the community does not necessarily guarantee access to or control of all the cloud's IT resources. Parties outside the community are generally not granted access unless allowed by the community.

9.7.3 Private Clouds

A private cloud is owned by a single organization. Private clouds enable an organization to use cloud computing technology as a means of centralizing access to IT resources by different parts, locations, or departments of the organization.

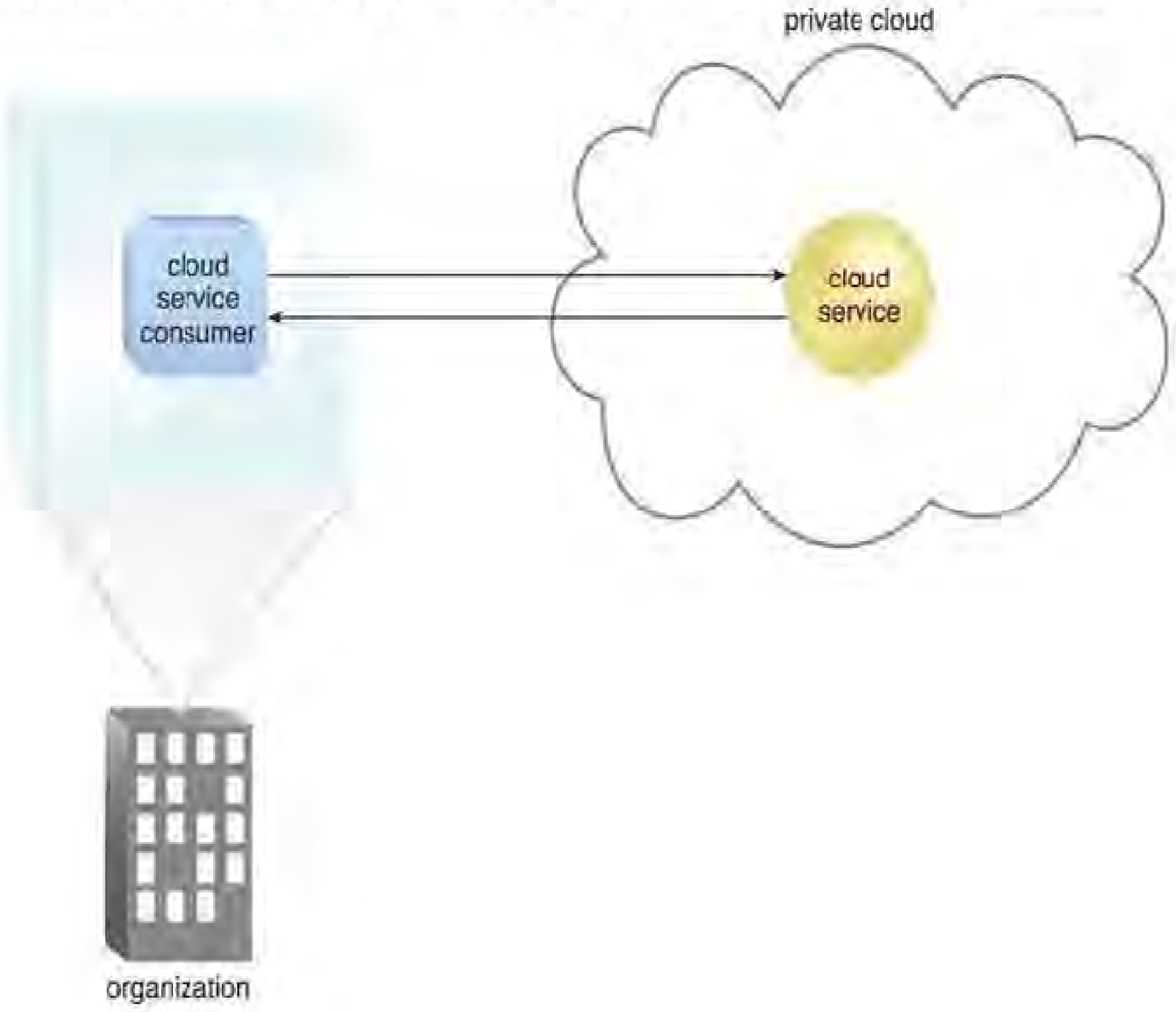


Figure 9.13 A cloud service consumer in the organization's on-premise environment accesses a cloud service hosted on the same organization's private cloud via a virtual private network.

The use of a private cloud can change how organizational and trust boundaries are defined and applied. The actual administration of a private cloud environment may be carried out by internal or outsourced staff.

With a private cloud, the same organization is technically both the cloud consumer and cloud provider (Figure 9.13).
In order to differentiate these roles:

- a separate organizational department typically assumes the responsibility for provisioning the cloud (and therefore assumes the cloud provider role)
- departments requiring access to the private cloud assume the cloud consumer role

It is important to use the terms "on-premise" and "cloud-based" correctly within the context of a private cloud. Even though the private cloud may physically reside on the organization's premises, IT resources it hosts are still considered "cloud-based" as long as they are made remotely accessible to cloud consumers. IT resources hosted outside of the private cloud by the departments acting as cloud consumers are therefore considered "on-premise" in relation to the private cloud-based IT resources.

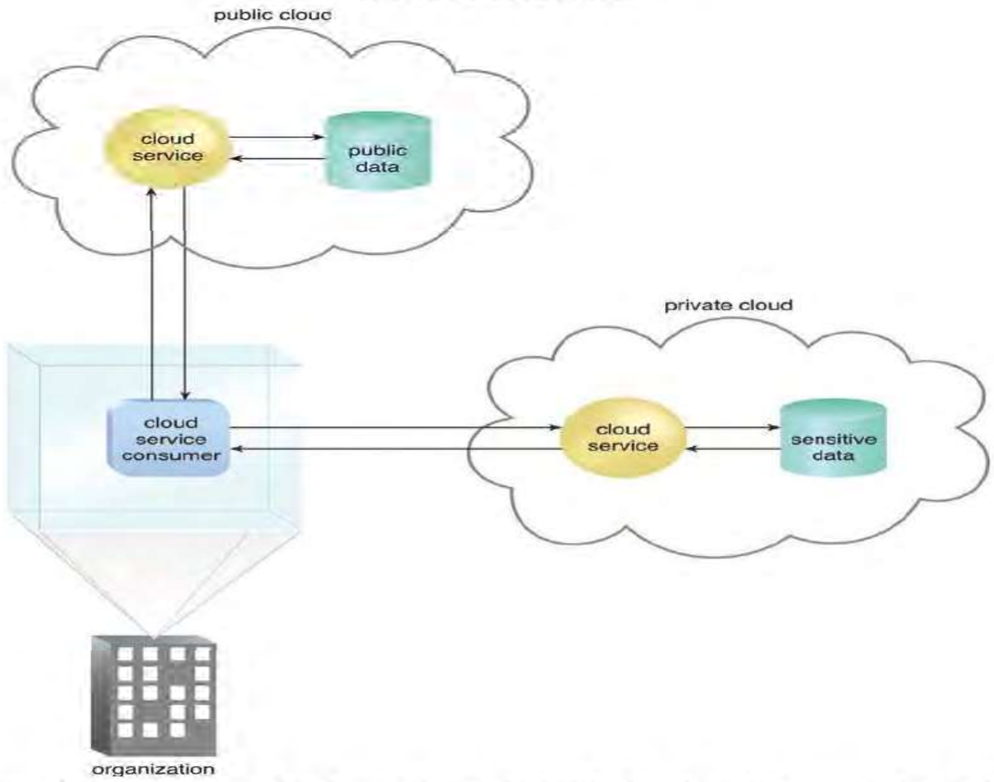


Figure 9.14 An organization using a hybrid cloud architecture that utilizes both a private and public cloud.

9.7.4 Hybrid Clouds

A hybrid cloud is a cloud environment comprised of two or more different cloud deployment models. For example, a cloud consumer may choose to deploy cloud services processing sensitive data to a private cloud and other, less sensitive cloud services to a public cloud. The result of this combination is a hybrid deployment model (Figure 9.14).

Hybrid deployment architectures can be complex and challenging to create and maintain due to the potential disparity in cloud environments and the fact that management responsibilities are typically split between the private cloud provider organization and the public cloud provider.

9.7.5 Other Cloud Deployment Models

Additional variations of the four base cloud deployment models can exist. Examples include:

- Virtual Private Cloud – Also known as a "dedicated cloud" or "hosted cloud," this model results in a self-contained cloud environment hosted and managed by a public cloud provider, and made available to a cloud consumer.
- Inter-Cloud – This model is based on an architecture comprised of two or more inter-connected clouds.

9.8 Summary

- Cloud Computing refers to both the applications delivered as services over the Internet and the scalable hardware and systems software that provide those services.
- The scalable hardware and software is known as a ***Cloud***.
- Virtualization is a fundamental part of cloud computing, especially in delivering Infrastructure as a Service.
- The person or organization that legally owns a cloud service is called a *cloud service owner.*

- The *cloud service consumer* is a temporary runtime role assumed by a software program when it accesses a cloud service.
- A cloud delivery model represents a specific, pre-packaged combination of IT resources offered by a cloud provider. Three common cloud delivery models are: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).
- A cloud deployment model represents a specific type of cloud environment, primarily distinguished by ownership, size, and access. There are four common cloud deployment models: Public cloud, Community cloud, Private cloud and Hybrid cloud.

9.9 Exercise

Q.1 Define cloud computing?

Q.2 What are the advantages of "Software As A Service" (SaaS)?

Q.3 Explain hybrid and community cloud.

Q.4 Explain the benefits of virtualization?

Q.5 How different cloud delivery models work? Explain in detail.

Q.6 Sketch the architecture of Cloud.

Q.7 Examine the reasons to adapt the cloud for upgraded internet applications and web services.

Q.8 Point out privacy key issues in cloud.

Q.9 Discuss in detail about cloud identity management.

Q.10 What are cloud provider and cloud broker?

References and Suggested Reading

1. Rajkumar Buyya, James Broberg, Andrzej Goscinski,"Cloud Computing Principles And Paradigms", Wiley Publications.
2. Zaigham Mahmood, Thomas Erl, Ricardo Puttini, "Cloud Computing: Concepts, Technology & Architecture", Prentice Hall

UNIT-10
MySQL in Web Development

Structure of the Unit

10.0 Objective

The objective of this unit is to show how PHP, JavaScript, MySQL works all together to make a web application. The aim of this unit is to makes student familiar with development of web application easily and efficiently. In this unit we shall focus upon the following:

- sorting of arrays, functions and string manipulation in PHP.
- Object Oriented PHP.
- Building your own Web application.

NOTE: You should have basic knowledge of HTML, CSS and JavaScript before reading this unit.

10.1 Introduction

This unit is focused on advance in PHP like sorting of array, string manipulation and function to reuse the existing code. At the end of the unit we will discuss on how to build our own web application by putting all together i.e. PHP, MySQL, jQuery, JavaScript etc.

10.2 PHP revisted: Sorting Data/Array and String Manipulation

(i) Sorting Data

In Unit-6, we studied the concept of array. In this section we discussed about some functions to sort the array according to their key/value in ascending or descending order. PHP provides some predefined function to sort the array, the description are as follows:

Function	**Description**	**Example**
sort()	sort the array in ascending order	sort($college)
rsort()	sort the array in descending order	**rsort(**$college**)**
asort()	sort the associative array in ascending order, according to their values	**asort(**$college**)**
ksort()	sort the associative array in ascending order, according to their Keys	**ksort(**$college**)**
arsort()	sort the associative array in descending order, according to their values	**arsort(**$college**)**
krsort()	sort the associative array in descending order, according to their keys	**arsort(**$college**)**

(ii) String Manipulation

As studied in Unit-6 string are the sequence of characters, PHP provides a set of in build functions to manipulate string. The description of some commonly used string manipulation functions are as follows:

Function	Description	Example
strlen()	returns the length of a string.	strlen("Hello world!"); // outputs 12
str_word_count()	counts the number of words in a string.	str_word_count("Hello world!"); // outputs 2
strrev()	reverses a string	strrev("Hello world!"); // outputs !dlrow olleH
strpos()	searches for a specific text within a string.	strpos("Hello world!", "world"); // outputs 6
str_replace()	replaces some characters with some other characters in a string.	str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!

10.3 Reusing Code and Writing Functions in PHP

A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value. Although there are many in built functions present in PHP like strlen() etc., in this section we studied to create and use our own function i.e. user defined function.

To create user defined function in PHP, you have to start with the word function:

```
function functionName() {
   code to be executed;
}
```

Example:

```
<?php
function writeMsg() {
   echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function.

Example:

```
<?php
       function addFunction($num1, $num2) {
         $sum = $num1 + $num2;
         echo "Sum of the two numbers is : $sum";
       }

       addFunction(10, 20);
    ?>
```

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

Example:

```
<?php
       function addFunction($num1, $num2) {
         $sum = $num1 + $num2;
         return $sum;
       }
       $return_value = addFunction(10, 20);

```

```
        echo "Returned value from the function : $return_value";
```

10.4 Object Oriented PHP

Object oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand. Object oriented concept includes class, object, data abstraction and encapsulation, inheritance, polymorphism, etc.

Object-oriented programming, with a limited feature set, is possible in PHP 4. With PHP 5, the object-oriented capabilities of PHP were greatly improved, with both more speed and added features.

In this section, we focus on overview of OOP concepts with respect to PHP.

Defining PHP classes

Class is a collection of objects of similar type. Classes are user-defined data type and behave like the build in types of a programming language.

We can define a new class in PHP as:

```
<?php
class myclass{
var $var1;
var $var2;
function myfun(){
        //function body
}
}
?>
```

Example:

```
<?php
class student{
```

```
var $scholar_number;
var $name;
function display(){
        //function definition
}
}
?>
```

Creating objects in PHP

Objects are run time entities in Object Oriented System. In PHP, object can be created using new operator as

```
$myobj = new myclass;
```

You can call the member functions related to that object as

```
$myobj -> myfun();
```

Constructor Function

Constructor is a special type of function which is called whenever any object is created. This function generally used to initializing the member variable.

You can define constructor using a special function __construct() in PHP as

```
function __construct($par1, $par2)
{
$this -> name=$par1;
$this->scholar_no=$par2;
}
```

NOTE: $this is a special type of variable provided in PHP which refers to the same object i.e. itself.

Destructor Function

This is also a special type of function which is used to release all the resources with-in a destructor.

Example:

```
class student{
function __destruct()
{
echo “Destructor called”;
}
}
```

Inheritance

We can share properties and methods between the classes using inheritance. Suppose we want child class to use the properties and methods of parent class then we can inherit parent class using extends keyword as:

```
class child extends parent {
//definition body
}
```

When child class inherit the parent class then it automatically has all member variable and member function i.e. we can access the parent class member variables and member functions using object of child class.

Function Overriding

If we are defining a method in the child class that has the same name, same arguments, and same return type as the method in the parent class. Then it is said to be overriding in PHP.

Example:

```
<?php
class parent {
  function myfun() {
    return "parent";
  }
}
```

```
class child extends parent {
  function myfun() {
    return "child";
  }
}

$childobj = new child;
$parentobj = new parent;
echo($childobj->myfun()); //"child"
echo($parentobj->myfun()); //"parent"
?>
```

Public, Private and Protected Members

PHP allows us to give control over the visibility of object properties and methods. There are three levels of visibility exists in PHP, they are public, protected and private.

If we prefix the class methods and properties with public keyword then this will set the corresponding class methods and properties to be "public" and allows a caller to manipulate them directly from the main body of the program. Public visibility is default visibility level for any class member under PHP.

If you want to make class members to be private or protected then you have to explicitly mark a particular property or method as private or protected. The "private" method and properties are only visible within the class that defines them, while the "protected" methods and properties are visible to both their base class and any child classes. Any attempt to access these methods and properties would result in a fatal error that can stop the script.

Example:

```
<?php
class parent{
public $name;
protected $age;
private $gender;
}
class child extends parent {
$p_obj = new parent;
$p_obj->name=”Neeraj”; //OK
$p_obj->age=29; // fatal error
$p_obj->gender=”Male”; // fatal error

$c_obj = new child;
$c_obj->name=”Arora”; // OK
$c_obj->age=13; // fatal error
$c_obj->gender=”Male”; // undefined


}
?>
```

Interfaces

Interface is an empty class which contains only the declaration of methods. So any class which implements this interface must contain the declared functions in it. So, interface is nothing but a strict ruling, which helps to extend any class and strictly implement all methods defined in interface. A class can use any interface by using the implements keyword. Please note that in interface you can only declare methods, but you cannot write their body. That means the body of all methods must remain blank.

Example:

```
interface Interface_Class{
public function myfun();
}
```

```
class simple_class implements Interface_Class{
//definition of myfun goes here;
}
```

Constants

Constant are the vaiable that can hold a value, which is immutable and does not change. It is declared with the keyword const. It is noted that the constant name does not proceed with $ sign as other variable do.

Example:

```
class student {
const max_marks=100;


function __constructor(){
//something in there;
}
}
```

Abstract Classes

Abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class. It is a design concept in program development and provides a base upon which other classes may be built.

Example:

```
abstract class Ab_class{
abstract function Ab_function(){
}
}
```

It is noted that the function definition inside the abstract class must be precedes with the keyword abstract.

10.5 Build your own PHP & MySQL Project Module

In this section, we build your own web application module with the help of PHP and MySQL. This web application has the functionality of log-in, log-out and has capable of create, read, update or delete (CRUD) records in database (MySQL).

1. Creating database and tables for the Project:-Create the following structure of tables using CREATE statements used in MySQL, as discussed earlier.

   Database Name: test

   Tables and their structure used:

   Table Name: users

user_id	int(11)
username	varchar(50)
password	varchar(50)
role	varchar(50)

   Table Name: test

test_id	bigint(20)
tname	varchar(20)

   Table Name: question

qid	int(11)
tid	int(11)
ques	varchar(5000)
option1	varchar(100)
option2	varchar(100)
option3	varchar(100)
option4	varchar(100)
ans	int(11)
descr	varchar(5000)

2. Creating the connection.php file:- This file contains all the code related to establish the connection between PHP and MySQL, the code is as follows:-

   connection.php

```
<?php
mysql_connect('localhost','root','');            /**            Syntax            ==>
mysql_connect(SERVERNAME,USERNAME,PASSWORD); **/
mysql_select_db('test'); /** This will select the databasename **/
?>
```

Later this file is included in all the related PHP file where we need to establish the connection using include_once( ) statement of PHP.

3. Home(index) Page:-

index.php

```
<?php
   include_once 'connection.php';
   session_start();

         if( $_SESSION['validation_status']!='OK' ):

                  header('Location: login.php');

         endif;

?>

<html>
   <head>
      <script src="jquery.js"/></script>
      <script src="jquery.validate.js"/></script>
      <meta  http-equiv="Content-Type"  content="text/html;  charset=UTF-
8">
      <meta charset="utf-8">
   <title>Select a Quiz</title>
   <!-- Sets initial viewport load and disables zooming  -->
   <meta  name="viewport"  content="initial-scale=1,  maximum-scale=1,
user-scalable=no">
   <!-- SmartAddon.com Verification -->
   <meta                                    name="smartaddon-verification"
content="936e8d43184bc47ef34e25e426c508fe" />
   <!-- site css -->
   <link rel="stylesheet" href="css/site.min.css">
   <link
href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,800
,700,400italic,600italic,700italic,800italic,300italic"         rel="stylesheet"
type="text/css">
   <!-- HTML5 shim, for IE6-8 support of HTML5 elements. All other JS at
the end of file. -->
   <!--[if lt IE 9]>
    <script src="js/html5shiv.js"></script>
    <script src="js/respond.min.js"></script>
   <![endif]-->

   <script type="text/javascript" src="js/site.min.js"></script>
     <title>Add a Question</title>
     <?php include_once 'navmenu.php';?>
   </head>
   <body>
<?php //echo $_GLOBALS['message'];?>
<div  class="panel"  style="width:  50%;  margin-left:  25%;  margin-right:
25%; margin-top: 10%">
         <div class="nav nav-tabs nav-justified">
          <ul id="myTab1" class="nav nav-tabs">
```

```
            <li  class="active"><a  href="#home3"  data-toggle="tab">Add
new Quiz</a></li>
            <li><a    href="#profile3"    data-toggle="tab">Select    Exiting
Quiz</a></li>
            </ul>
           <div id="myTabContent" class="tab-content">
            <div class="tab-pane fade active in" id="home3">


               <p>
               <div class="panel panel-info">
                <div class="panel-heading">
                   <h1 class="panel-title">Add a question</h1>
                </div>
   <div class="panel-body">
      <form id="form1"  method="post" action="insert-quiz.php">
                <label      class="description">Enter      your      Quiz-Name
</label><br>
                <div>
                         <input       name="tname"       class="form-control"
type="text" required>
                </div>
                <input   id="saveForm"   class="btn   btn-info   navbar-btn"
type="submit" name="submit" value="Submit" />


      </form>
   </div>
              </div>


            </p>
            </div>
            <div class="tab-pane fade" id="profile3">
             <p>
              <div class="panel panel-info">
               <div class="panel-heading">
                 <h1 class="panel-title">Select from Existing Quiz</h1>
               </div>
  <div class="panel-body">
    <form id="form1"  method="post" action="insert.php">



                   <table class="table">
        <thead>
         <tr>
          <th>S. No.</th>
          <th>Quiz-Name</th>
          <th>Add Question</th>
          <th>Show Question</th>
          <th>Delete Quiz</th>
         </tr>
        </thead>
        <tbody>
                    <?php
              mysql_connect('localhost','root','');    /**    Syntax    ==>
mysql_connect(SERVERNAME,USERNAME,PASSWORD); **/
                mysql_select_db('test');   /**   This   will   select   the
databasename **/

              $result = mysql_query("SELECT * FROM test");
```

```
                              echo                                    "<p
style='color=red'>".mysql_error()."</p>";
                              $sno=1;
                              while($row = mysql_fetch_array($result) )
                         {?>
          <tr>
           <th><?php echo $sno;?></th>
           <td><?php echo $row["tname"] ?></td>
           <td><a  class='btn  btn-default'  href='add-ques.php?tid=<?php
echo $row["tid"] ?>'>Add</a><?php echo ""?></td>
           <td><a       class='btn       btn-warning'       href='show-ques-
normal.php?tid=<?php    echo    $row["tid"]    ?>'>Show</a><?php    echo
""?></td>
           <td><a  class='btn  btn-danger'  href='deleteById.php?tid=<?php
echo $row["tid"] ?>'>Delete</a><?php echo ""?></td>
              </tr>
                      <?php
                   $sno++;}


                      ?>
         </tbody>
              </table>


     </form>
  </div>
             </div>


           </p>
           </div>
```

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

4. Login, register (as a new user) and Logout from user account:-

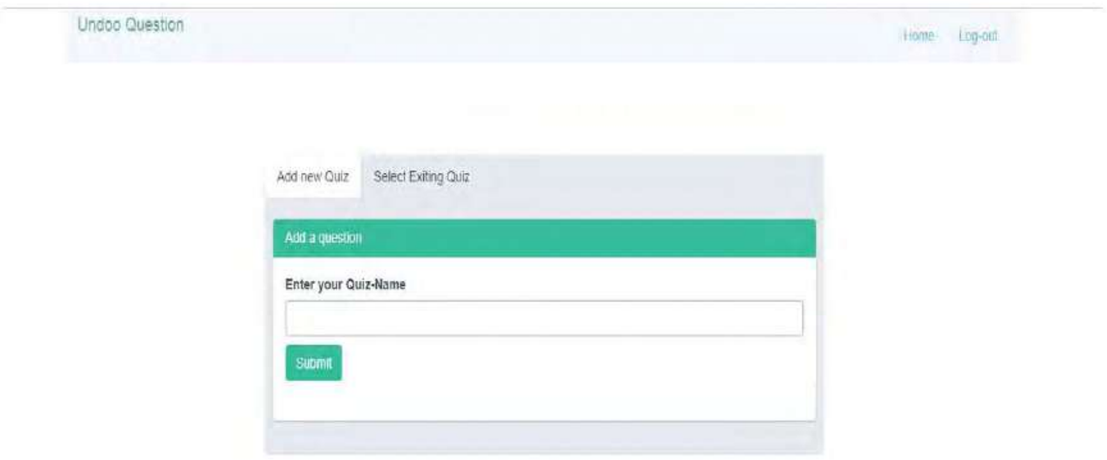
login.php

```
<html>
  <head>
    <script src="jquery.js"/></script>
    <script src="jquery.validate.js"/></script>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
    <meta charset="utf-8">
  <title>Select a Quiz</title>
  <!-- Sets initial viewport load and disables zooming -->
  <meta name="viewport" content="initial-scale=1, maximum-scale=1,
user-scalable=no">
  <!-- SmartAddon.com Verification -->
  <meta name="smartaddon-verification"
content="936e8d43184bc47ef34e25e426c508fe" />
  <!-- site css -->
  <link rel="stylesheet" href="css/site.min.css">
  <link
href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,800
,700,400italic,600italic,700italic,800italic,300italic" rel="stylesheet"
type="text/css">
```

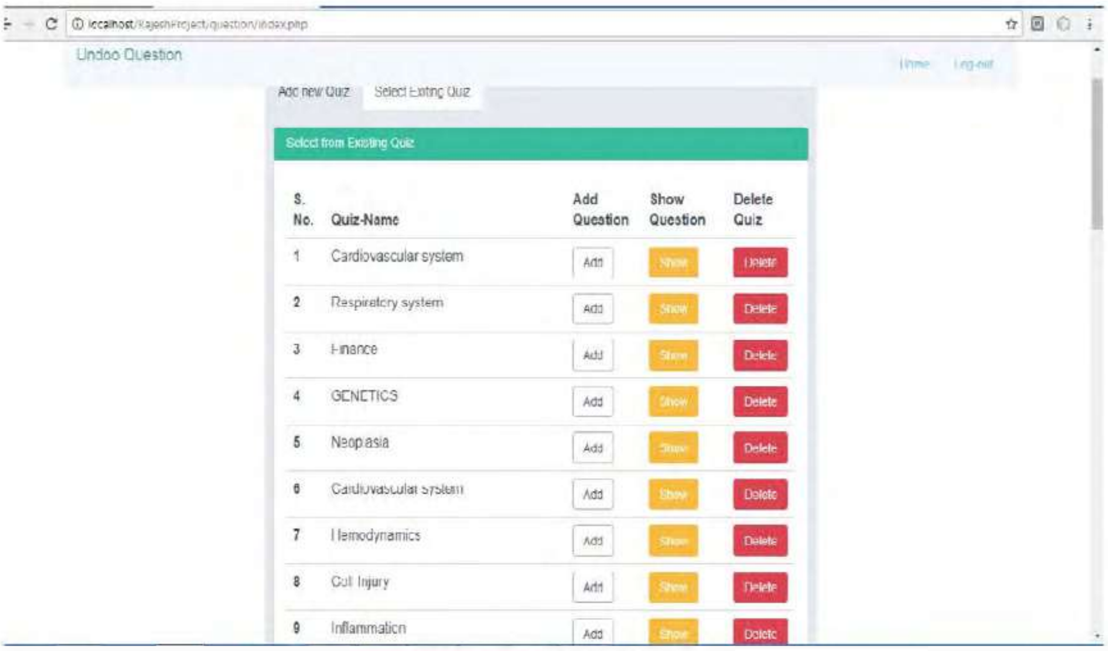
```
   <!-- HTML5 shim, for IE6-8 support of HTML5 elements. All other JS at
the end of file. -->
   <!--[if lt IE 9]>
     <script src="js/html5shiv.js"></script>
     <script src="js/respond.min.js"></script>
   <![endif]-->
   <script type="text/javascript" src="js/site.min.js"></script>
   <?php include_once 'navmenu.php';?>
      <title>Add a Question</title>
   </head>
   <body>
         <?php echo $_GLOBALS['message'];?>
    <div class="panel panel-danger" style="width: 40%; margin-top: 10%;
margin-left: 5%; float:left;" >
                  <div class="panel-heading">
                     <h1 class="panel-title">Enter to Log-in</h1>
                  </div>
   <div class="panel-body">
<form  method="post" action="do_login.php">


                  <label class="description" >User Name </label><br>
                  <div>
                          <input     class="form-control"     name="username"
type="text" required>
                  </div>
                  <br><br>
                  <label class="description">Password</label><br>
            <div>
```

```
                    <input     class="form-control"     name="password"
type="password" required>
                </div>
            <br/>
                <input  id="saveForm"  class="btn  btn-danger  btn-block"
type="submit" name="submit" value="Log-in" />

</form>
    </div>
        </div>

<div  class="panel  panel-success"  style="width:  48%;  margin-top:  10%;
margin-right: .5%; float:right;" >
                <div class="panel-heading">
                   <h1 class="panel-title">New user Registration</h1>
                </div>
  <div class="panel-body">
<form  method="post" action="do_register.php">



                <label class="description" >User Name </label><br>
                <div>
                    <input     class="form-control"     name="username"
type="text" required>
                </div>

                <label class="description">Password</label><br>
            <div>
```

```
                        <input      class="form-control"      name="password"
type="password" required>
                </div>
                <label class="description">Re-Type Password</label><br>
            <div>
                        <input     class="form-control"     name="repassword"
type="password" required>
                </div>


                <label class="description">E-mail id</label><br>
            <div>
                        <input        class="form-control"        name="email"
type="text" required>
                </div>
                <label class="description">Contact Number</label><br>
            <div>
                        <input       class="form-control"       name="contact"
type="text" required>
                </div>
                <label class="description">Address</label><br>
            <div>
                        <input       class="form-control"       name="address"
type="text" required>
                </div>
                <label class="description">City</label><br>
            <div>
                        <input class="form-control" name="city" type="text"
required>
                </div>
                <label class="description">Pincode</label><br>
            <div>
                        <input   class="form-control"   name="pincode"
type="text" required>
                </div></br>
                <input id="saveForm" class="btn btn-success btn-block"
type="submit" name="submit" value="Register" />

</form>
    </div>
        </div>
</body>
</html>
<?php
session_start();

if( isset($_SESSION['validation_status']) ):

        if( $_SESSION['validation_status']=='OK' ):

                header('Location: index.php');

        endif;

endif;

?>
```



Enter your Quiz-Name



localhost/RajeshProject/question/index.php

S. No.	Quiz-Name	Add Question	Show Question	Delete Quiz
1	Cardiovascular system	Add	Show	Delete
2	Respiratory system	Add	Show	Delete
3	Finance	Add	Show	Delete
4	GENETICS	Add	Show	Delete
5	Neoplasia	Add	Show	Delete
6	Cardiovascular system	Add	Show	Delete
7	Hemodynamics	Add	Show	Delete
8	Cell Injury	Add	Show	Delete
9	Inflammation	Add	Show	Delete

Figure: Home Page

<u>logout.php:</u>

```
<?php
session_start();

unset($_SESSION['validation_status']);
```

```
header('Location: login.php');

?>
```

dologin.php

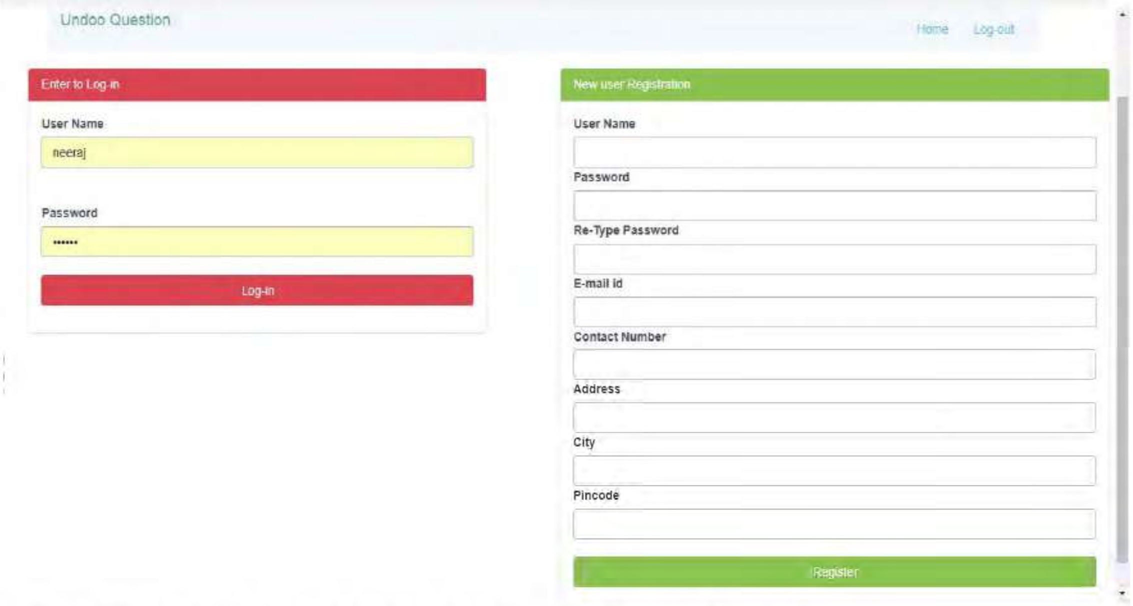
```
<?php
include_once 'connection.php';

        $user = $_POST['username'];
        $pass = $_POST['password'];

        $query = mysql_query("SELECT * from users WHERE username =
'$user' AND password = '$pass'") or die( mysql_error() );

                $count_record = mysql_num_rows( $query );

                if($count_record > 0):

                        session_start();

                        $_SESSION['validation_status'] = 'OK';
                        $_GLOBALS['message']="Login Successful";
                        header('Location: index.php');
                else:
                        $_GLOBALS['message']="User Name or Password
are not match";
                        header('Location: login.php');

                endif;
```

```
?>
```

doregister.php

```
<?php
include_once 'connection.php';


        $user = $_POST['username'];
        $pass = $_POST['password'];
        $repassword = $_POST['repassword'];
        $email = $_POST['email'];
        $contact = $_POST['contact'];
        $address = $_POST['address'];
        $city = $_POST['city'];
        $pincode = $_POST['pincode'];
        $role="student";
$query = mysql_query("select username from users where
username='$user'"); //checking for user name

if($_POST['password']!=$_POST['repassword'])
{
//password is not match
$_GLOBALS['message']="Password is not match";
//echo "not match";exit;
}
else if(mysql_num_rows($query)>0)//User name already exists
{
$_GLOBALS['message']="User name already exists";
header('Location: index.php');
}
else
```

```
{
        //echo "insert into users values('$user','$pass','$repassword','$email',$contact,'$address','$city',$pincode)";exit;
        $query = mysql_query("insert into users values('','$user','$pass','$email','$contact','$address','$city','$pincode','$role');
") or die( mysql_error() );
        $_GLOBALS['message']="Account created successfully <a href=\"index.php\">Here</a> to LogIn";
        header('Location: index.php');


}
?>
```



5. Inserting new quiz and new question in a particular quiz:

<u>insert-quiz.php</u>

```
<?php
include_once 'connection.php';
$sql="INSERT INTO test (tname)
VALUES
```

```
('$_POST[tname]')";
//echo $sql;exit;
if (!mysql_query($sql))
  {
  die('Error: ' . mysql_error());
  }
echo '<div class="alert alert-success alert-dismissable" style="width: 90%;
margin-left: 5%; margin-right: 5%; margin-top: 5%">
<button class="close" aria-hidden="true" data-dismiss="alert"
type="button">×</button>
<strong>Well Done</strong>
New Quiz Added
</div>';


mysql_close();
include_once 'index.php';
?>
```

insert-ques.php

```
<?php
include_once 'connection.php';


$sql="INSERT INTO question (tid,ques, option1, option2, option3,
option4, ans, discr)
VALUES
('$_POST[tid]','$_POST[ques]','$_POST[option1]','$_POST[option2]','$_P
OST[option3]','$_POST[option4]','$_POST[ans]','$_POST[discr]')";
echo $sql;
if (!mysql_query($sql))
  {
```

```
 die('Error: ' . mysql_error());
 }?>
<div class="alert alert-warning alert-dismissable" style="width: 90%; margin-left: 5%; margin-right: 5%; margin-top: 5%">
            <button class="close" aria-hidden="true" data-dismiss="alert" type="button">×</button>
            <strong>Well Done</strong>
            Question is Added
         </div>
<?php
mysql_close();

header("Location: add-ques.php?tid=". $_POST["tid"]);

?>
```

6. Showing the available questions in a particular quiz:

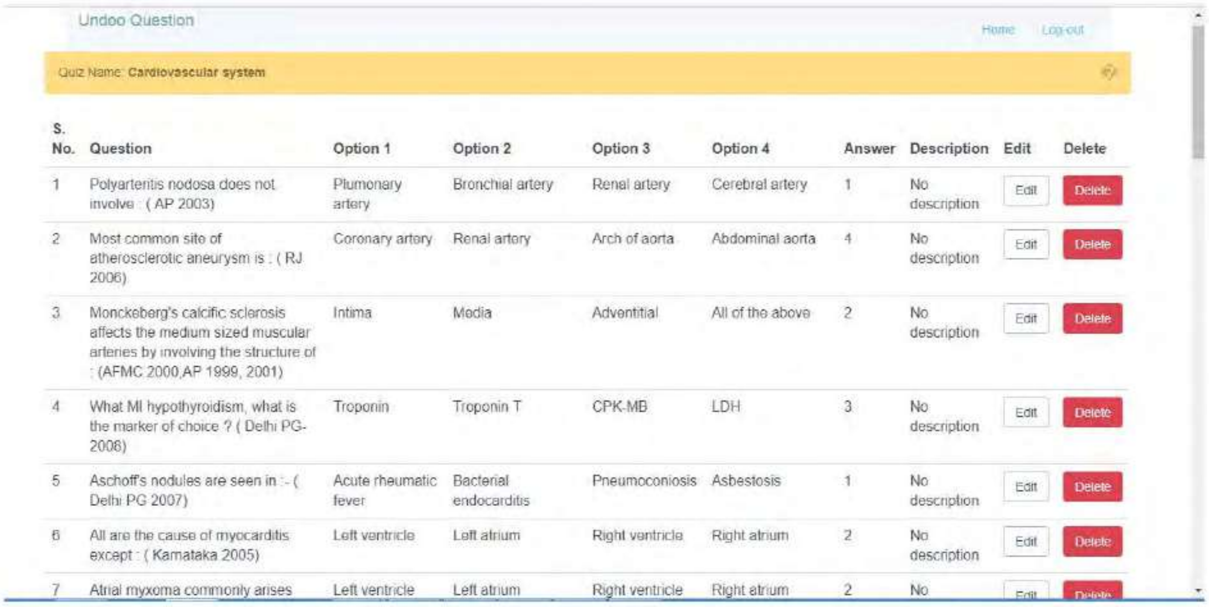
show-ques-normal.php

```
<?php
   include_once 'connection.php';
   session_start();

         if( $_SESSION['validation_status']!='OK' ):

                  header('Location: login.php');

         endif;

?>
```

```
<html>
   <head>
      <script src="jquery.js"/></script>
      <script src="jquery.validate.js"/></script>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
      <meta charset="utf-8">
   <title>List of Question</title>
   <!-- Sets initial viewport load and disables zooming  -->
   <meta name="viewport" content="initial-scale=1, maximum-scale=1,
user-scalable=no">
   <!-- SmartAddon.com Verification -->
   <meta                                   name="smartaddon-verification"
content="936e8d43184bc47ef34e25e426c508fe" />
   <!-- site css -->
   <link rel="stylesheet" href="css/site.min.css">
   <link
href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,800
,700,400italic,600italic,700italic,800italic,300italic"         rel="stylesheet"
type="text/css">
   <!-- HTML5 shim, for IE6-8 support of HTML5 elements. All other JS at
the end of file. -->
   <!--[if lt IE 9]>
     <script src="js/html5shiv.js"></script>
     <script src="js/respond.min.js"></script>
   <![endif]-->

   <script type="text/javascript" src="js/site.min.js"></script>
      <title>List of Questions</title>
```

```
    <?php include_once 'navmenu.php';?>
  </head>
  <body style="width: 90%; margin-left: 5%; margin-right: 5%; margin-
top: 5%">

<div class="alert alert-warning alert-dismissable" >
        <button class="close" aria-hidden="true" data-dismiss="alert"
type="button">×</button>
        Quiz Name:
        <strong> <?php
        mysql_connect('localhost','root','');        /**     Syntax     ==>
mysql_connect(SERVERNAME,USERNAME,PASSWORD); **/
          mysql_select_db('test'); /** This will select the databasename
**/
        $result = mysql_query("SELECT tname FROM test where
tid=".$_GET['tid']);
        $row = mysql_fetch_array($result);
        echo $row['tname'];
        ?></strong>

      </div>
<table class="table">
       <thead>
        <tr>
         <th>S. No.</th>
         <th >Question</th>
         <th>Option 1</th>
         <th>Option 2</th>
         <th>Option 3</th>
```

```
              <th>Option 4</th>
              <th>Answer</th>
              <th>Description</th>
              <th>Edit</th>
              <th>Delete</th>
            </tr>
          </thead>
          <tbody>
              <?php
include_once 'connection.php';

$result  =  mysql_query("SELECT  *  FROM  question  where  tid  =
".$_GET['tid']);
 $i=1;
while($row = mysql_fetch_array($result))
  {

  // $qid=$row["qid"];
   //$tid=$row["tid"];
   $question=$row["ques"];
   $answer=$row["ans"];
   $option1=$row["option1"];
   $option2=$row["option2"];
   $option3=$row["option3"];
   $option4=$row["option4"];

?>
            <tr>
              <td><?php echo $i; ?></td>
            <td><?php echo $question; ?></td>
            <td><?php echo $option1; ?></td>
            <td><?php echo $option2;?></td>
            <td><?php echo $option3;?></td>
            <td><?php echo $option4; ?></td>
            <td><?php echo $answer; ?></td>
            <td><?php echo 'No description';?></td>
            <td><a class='btn btn-default' href='edit-ques.php?tid=<?php
echo $row["tid"] ?>&qid=<?php echo $row['qid'];?>'>Edit</a></td>
            <td><a class='btn btn-danger' href='del-ques.php?tid=<?php
echo $row["tid"] ?>&qid=<?php echo $row['qid'];?>'>Delete</a></td>
           </tr>
 <?php $i++;}?>
          </tbody>
                </table>
      <a class="btn btn-warning" href="show-ques-html.php?tid=<?php
echo $_GET['tid'];?>">Show Question(HTML View)</a>
</body>
</html>
```



Quiz Name: **Cardiovascular system**

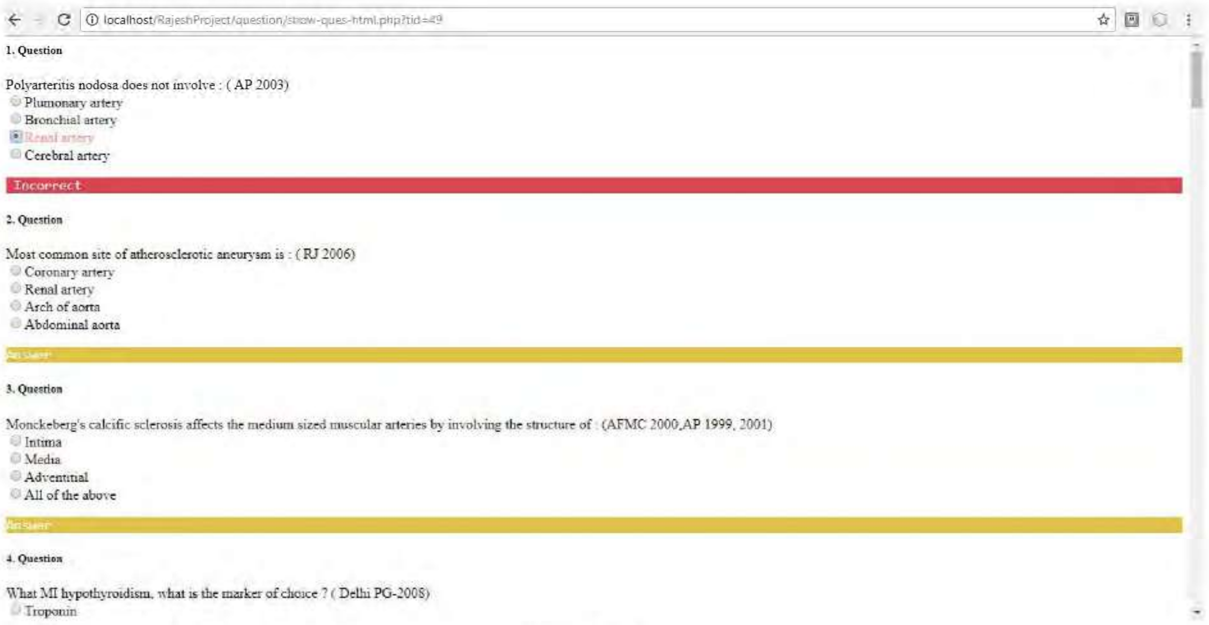
S. No.	Question	Option 1	Option 2	Option 3	Option 4	Answer	Description	Edit	Delete
1	Polyarteritis nodosa does not involve : (AP 2003)	Plumonary artery	Bronchial artery	Renal artery	Cerebral artery	1	No description	Edit	Delete
2	Most common site of atherosclerotic aneurysm is : (RJ 2006)	Coronary artery	Renal artory	Arch of aorta	Abdominal aorta	4	No description	Edit	Delete
3	Monckeberg's calcific sclerosis affects the medium sized muscular arteries by involving the structure of : (AFMC 2000,AP 1999, 2001)	Intima	Media	Adventitial	All of the above	2	No description	Edit	Delete
4	What MI hypothyroidism, what is the marker of choice ? (Delhi PG-2008)	Troponin	Troponin T	CPK-MB	LDH	3	No description	Edit	Delete
5	Aschoff's nodules are seen in :- (Delhi PG 2007)	Acute rheumatic fever	Bacterial endocarditis	Pneumoconiosis	Asbestosis	1	No description	Edit	Delete
6	All are the cause of myocarditis except : (Karnataka 2005)	Left ventricle	Left atrium	Right ventricle	Right atrium	2	No description	Edit	Delete
7	Atrial myxoma commonly arises	Left ventricle	Left atrium	Right ventricle	Right atrium	2	No	Edit	Delete

show-ques-html.php

```
<?php
include_once 'connection.php';


$result = mysql_query("SELECT * FROM question where tid = ".$_GET['tid']);
 $i=0;
while($row = mysql_fetch_array($result))
 {
   $qid=$row["qid"];
  $tid=$row["tid"];
  $question=$row["ques"];
  $answer=$row["ans"];
  $option1=$row["option1"];
  $option2=$row["option2"];
  $option3=$row["option3"];
  $option4=$row["option4"];
?>
  <h5><?php echo ++$i.'. ';?> Question</h5>
```

```
<div>
<div>
<?php echo $question;?>
</div>


        <div  onclick="this.style.color=' <?php if($answer == 1) {echo '#6db46d';} else {echo '#ff9191';} ?>'"><label><input onclick="<?php if($answer == 1) {echo 'correct';} else {echo 'wrong';} ?>(this.name);" type="radio"  name="<?php echo $qid.'_'.$tid;?>" value="1" /><?php echo $option1; ?></label></div>
        <div  onclick="this.style.color=' <?php if($answer == 2) {echo '#6db46d';} else {echo '#ff9191';} ?>'"><label><input onclick="<?php if($answer == 2) {echo 'correct';} else {echo 'wrong';} ?>(this.name);" type="radio"  name="<?php echo $qid.'_'.$tid;?>" value="2" /><?php echo $option2; ?></label></div>
   <div  onclick="this.style.color=' <?php if($answer == 3) {echo '#6db46d';} else {echo '#ff9191';} ?>'"><label><input onclick="<?php if($answer == 3) {echo 'correct';} else {echo 'wrong';} ?>(this.name);" type="radio"  name="<?php echo $qid.'_'.$tid;?>" value="3" /><?php echo $option3; ?></label></div>
   <div  onclick="this.style.color=' <?php if($answer == 4) {echo '#6db46d';} else {echo '#ff9191';} ?>'"><label><input onclick="<?php if($answer == 4) {echo 'correct';} else {echo 'wrong';} ?>(this.name);" type="radio"  name="<?php echo $qid.'_'.$tid;?>" value="4" /><?php echo $option4; ?></label></div>


</div>
<div>
<pre style="background: #DEC349; color: #FFFFFF; font-size: larger"id="<?php echo $qid.'_'.$tid;?>">Answer</pre>
</div>
<div></div>
```

```
<?php

if($i%5==0)
{?>
<script type="text/javascript">// <![CDATA[
function correct(qid)
{
document.getElementById(qid).innerHTML="<span style='color: #FFFFFF;'>
Correct </span>";
document.getElementById(qid).style.background="#3C763D";
document.getElementById(qid).style.fontSize="larger";


}
function wrong(qid)
{
document.getElementById(qid).innerHTML="<span style='color: #FFFFFF;'>
Incorrect </span>";
document.getElementById(qid).style.background="#DA4453";
document.getElementById(qid).style.fontSize="larger";


}
// ]]></script>
<!--nextpage-->
<?php }
 }
mysql_close();
?>
 <script type="text/javascript">// <![CDATA[
function correct(qid)
{
document.getElementById(qid).innerHTML="<span    style='color:    #FFFFFF;'>
Correct </span>";
document.getElementById(qid).style.background="#3C763D";
document.getElementById(qid).style.fontSize="larger";
}
function wrong(qid)
{
document.getElementById(qid).innerHTML="<span    style='color:    #FFFFFF;'>
Incorrect </span>";
document.getElementById(qid).style.background="#DA4453";
document.getElementById(qid).style.fontSize="larger";
}
// ]]></script>
```



localhost/RajeshProject/question/show-ques-html.php?tid=49

1. Question

Polyarteritis nodosa does not involve : (AP 2003)
- Plumonary artery
- Bronchial artery
- [✓ Renal artery]
- Cerebral artery

Incorrect

2. Question

Most common site of atherosclerotic aneurysm is : (RJ 2006)
- Coronary artery
- Renal artery
- Arch of aorta
- Abdominal aorta

Answer

3. Question

Monckeberg's calcific sclerosis affects the medium sized muscular arteries by involving the structure of : (AFMC 2000,AP 1999, 2001)
- Intima
- Media
- Adventitial
- All of the above

Answer

4. Question

What MI hypothyroidism, what is the marker of choice ? (Delhi PG-2008)
- Troponin

7. Update the questions in a particular quiz:
   edit-ques.php

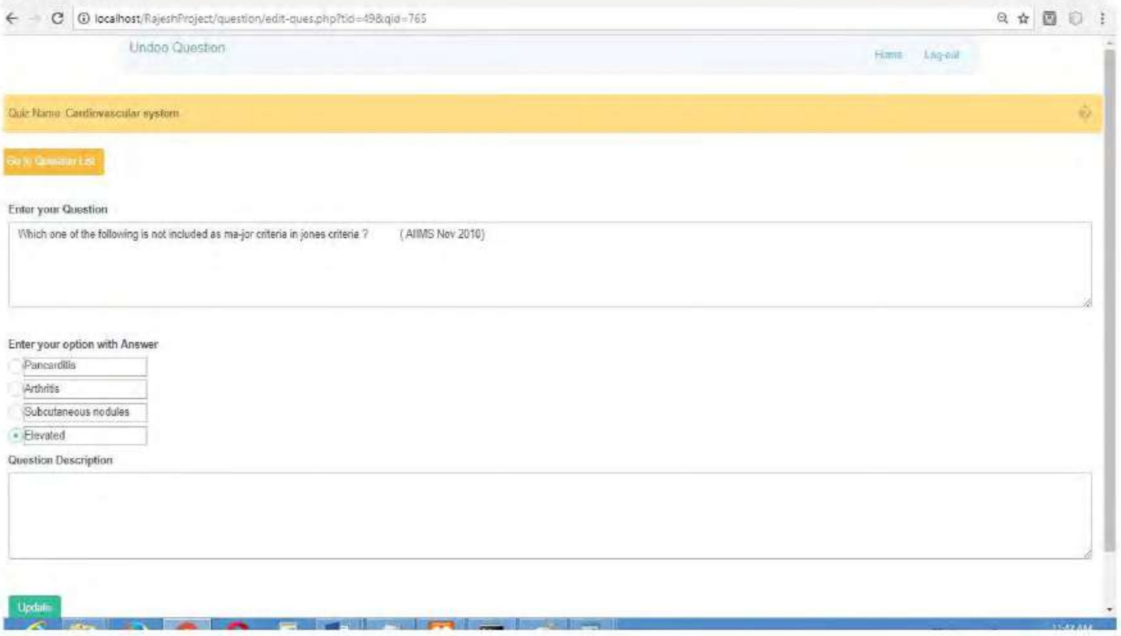
```
<script type="text/javascript" src="<./js/tinymce/tinymce.min.js"></script>
```

```
<script type="text/javascript">
tinymce.init({
   selector: "textarea",
   plugins: [
      "advlist autolink lists link image charmap print preview anchor",
      "searchreplace visualblocks code fullscreen",
      "insertdatetime media table contextmenu paste moxiemanager"
   ],
   toolbar: "insertfile undo redo | styleselect | bold italic | alignleft
aligncenter alignright alignjustify | bullist numlist outdent indent | link
image"
});
</script>
<?php
                    include    'connection.php';    /**    calling    of
connection.php that has the connection code **/

                    $qid = $_GET['qid']; /** get the student ID **/
                    $tid = $_GET['tid'];
                    if( isset( $_POST['update'] ) ): /** A trigger that
execute after clicking the submit      button **/

                           /*** Putting all the data from text into
variables **/
                           $ques = $_POST['ques'];
                           $option1 = $_POST['option1'];
                           $option2 = $_POST['option2'];
                           $option3 = $_POST['option3'];
                           $option4 = $_POST['option4'];
                           $ans = $_POST['ans'];
                           $discr = $_POST['discr'];
```

```
                                    mysql_query("UPDATE question SET ques =
'$ques', option1 = '$option1', option2 = '$option2', option3 = '$option3',
option4 = '$option4', ans = '$ans', discr = '$discr' WHERE qid = '$qid'")
                                                    or die(mysql_error());
/*** execute the insert sql code **/

                                    echo "<div class='alert alert-success alert-
dismissable'> <button class='close' aria-hidden='true' data-dismiss='alert'
type='button'>×</button> Successfully Updated. </div>"; /** success
message **/
                        //header("Location: show-ques-normal.php?tid=".
$_POST["tid"]);
                        endif;


                        $result = mysql_query("SELECT * FROM question
WHERE qid='$qid'");

                        $data = mysql_fetch_array( $result );
                ?>
<html>
   <head>
  <script src="jquery.js"/></script>
      <script src="jquery.validate.js"/></script>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
      <meta charset="utf-8">
   <title>Edit a Question</title>
   <!-- Sets initial viewport load and disables zooming -->
   <meta name="viewport" content="initial-scale=1, maximum-scale=1,
user-scalable=no">
   <!-- SmartAddon.com Verification -->
   <meta name="smartaddon-verification"
content="936e8d43184bc47ef34e25e426c508fe" />
```

```
   <!-- site css -->
   <link rel="stylesheet" href="css/site.min.css">
   <link
href="http://fonts.googleapis.com/css?family=Open+Sans:400,300,600,800
,700,400italic,600italic,700italic,800italic,300italic"          rel="stylesheet"
type="text/css">
   <!-- HTML5 shim, for IE6-8 support of HTML5 elements. All other JS at
the end of file. -->
   <!--[if lt IE 9]>
    <script src="js/html5shiv.js"></script>
    <script src="js/respond.min.js"></script>
   <![endif]-->
   <script type="text/javascript" src="js/site.min.js"></script>
   <?php include_once 'navmenu.php';?>
   </head>

<body style="margin-top: 5%">
   <div class="alert alert-warning alert-dismissable" >
            <button   class="close"   aria-hidden="true"   data-dismiss="alert"
type="button">×</button>
            Quiz Name:
            <strong> <?php
            mysql_connect('localhost','root','');       /**       Syntax       ==>
mysql_connect(SERVERNAME,USERNAME,PASSWORD); **/
              mysql_select_db('test_plugin');  /**  This  will  select  the
databasename **/
            $result  =  mysql_query("SELECT  tname  FROM  test  where
tid=".$tid);
            $row = mysql_fetch_array($result);
            echo $row['tname'];
            ?></strong>

        </div>
```

```
  <a   class="btn   btn-warning"   href="show-ques-normal.php?tid=<?php
echo $_GET['tid'];?>">Go to Question List</a>
<div class="panel-body"> <form id="form1"  method="post" action="">

    <input   type="text"   name="tid"   required   value="<?php   echo
$_GET['tid'];?>" hidden><br>
            <label       class="description">Enter       your       Question
</label><br>
            <div>
                  <textarea     class     ="form-control"     id="ques"
name="ques" class=ques"  rows="5" cols="40" required=""><?php echo
$data['ques'] ?></textarea>
            </div>
            <br><br>
            <label class="description">Enter your option with Answer
</label><br>

      <div class="radio">
            <input id="flat-radio-1" type="radio" name="ans" value="1"
<?php if($data['ans']==1) {echo'checked';} ?>  required><input type="text"
name="option1" value="<?php echo $data['option1'] ?>" required><br>
      </div>

      <div class="radio">
      <input class ="iCheck-helper" type="radio" name="ans" value="2"
<?php if($data['ans']==2) {echo'checked';}?> required><input type="text"
name="option2" value="<?php echo $data['option2'] ?>" required><br>
      </div>

      <div class="radio">
        <input    class    ="iCheck-helper"    type="radio"    name="ans"
value="3"  <?php  if($data['ans']==3)  {echo'checked';}?>  required><input
type="text"   name="option3"   value="<?php   echo   $data['option3']   ?>"
required><br>
```

```
        </div>
        <div class="radio">
            <input class ="iCheck-helper" type="radio" name="ans"
value="4" <?php if($data['ans']==4) {echo'checked';}?> required><input
type="text" name="option4" value="<?php echo $data['option4'] ?>"
required><br>
        </div>
            <label class="description">Question Description </label><br>
                <div>
                        <textarea class ="form-control" id="discr"
name="discr" class=discr" rows="5" cols="40"><?php echo $data['discr']
?></textarea>
                </div>
                <br><br>
                <input id="saveForm" class="btn btn-info navbar-btn"
type="submit" name="update" value="Update" />

</form>        </div>
   <a class="btn btn-warning" href="show-ques-normal.php?tid=<?php
echo $_GET['tid'];?>">Go to Question List</a>
</body>
</html>
```

localhost/RajeshProject/question/edit-ques.php?tid=49&qid=765

Undoo Question

Quiz Name: Cardiovascular system

Enter your Question

Which one of the following is not included as ma-jor criteria in jones criteria ? (AIIMS Nov 2010)

Enter your option with Answer

- ○ Pancarditis
- ○ Arthritis
- ○ Subcutaneous nodules
- ● Elevated

Question Description

8. Deleting the questions:-

del-ques.php

```
<?php

include_once "connection.php"; /** calling of connection.php that has the
connection code **/
mysql_query("DELETE FROM question where qid =".$_GET['qid']);
mysql_close();

header("Location: show-ques-normal.php?tid=".$_GET['tid']);

?>
```

10.6 Answer to Self Learning Exercise

Q.1 For running the SQL queries PHP uses:

a) mysql_query()

b) mysql_fetch()

c) mysql_run()

d) All of the above

Q.2 $.post method is used to:

a) Everyone can read, group can execute only and the owner can read and Write.

b) Everyone can read and write, but owner alone can execute,

c) request data from the server using an HTTP POST request.

d) Everyone can read and write and execute.

Q.3 CRUD stands for:

a) Control Read Understand Delete.

b) Create, Read, Update Delete.

c) Both a and b.

d) None of the above.

10.7 Summary

- PHP, JavaScript, MySQL works all together to make a web application easily and efficiently.
- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.
- PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like.
- Class is a collection of objects of similar type. Classes are user-defined data type and behave like the build in types of a programming language.
- PHP allows us to give control over the visibility of object properties and methods.
- Interface is an empty class which contains only the declaration of methods.
- Constant are the vaiable that can hold a value, which is immutable and does not change.
- We build your own web application module with the help of PHP and MySQL.
- This web application has the functionality of log-in, log-out and has capable of create, read, update or delete (CRUD) records in database (MySQL).

10.8 Answers to Self-Learning Exercise

Q.1 (d)

Q.2 (c)

Q.3 (b)

10.9 Exercise

Q. 1 Write short note on Object Oriented PHP.

Q. 2 How code is reused with the help of functions in PHP? Explain.

Q. 3 Write an user defined function to sort an array elemets in PHP:

Q. 4 Explain sorting of data and string manipulation functions in detail.

Q. 5 How to create an user defined function in PHP? Explain with an example.

Q.6 Explain the following with one example

(a) class in PHP

(b) constructor function

References and Suggested Readings

1. Matt West, HTML5 Foundations, John Wiley and Sons, Ltd, Publication, First Edition.
2. Steve Suehring and Janet Valade, PHP, MySQL®, JavaScript & HTML5 All-in-One For Dummies, John Wiley & Sons Publication.
3. Beginning PHP and MySQL: From Novice to Professional 4th Edition by W. Jason Gilmore, Apress
4. PHP: The Complete Reference 1st Edition, by Steven Holzner, Tata Mcgraw Hill Education Private Limited
5. http://www.php.net/manual/en/
6. www.w3schools.com.
7. www.tutorialpoint.com.

UNIT-11
jQuery

Structure of the Unit

11.0 Objective

The objective of this unit is to make the students familiar with jQuery so that student makes web application easily and efficiently. In this unit we shall focus upon the following:

Why and where to use jQuery.

- Features of jQuery.
- jQuery Event Methods
- jQuery Effects
- jQuery – AJAX Methods.

NOTE: You should have basic knowledge of HTML, CSS and JavaScript before reading this unit.

11.1 Introduction

jQuery is an open source JavaScript library that simplifies the interactions between an HTML document, or more precisely the Document Object Model (aka the DOM), and JavaScript. jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

jQuery
write less, do more.

Figure11.1: jQuery Official Logo

Features of jQuery:

- It's open source, and the project is licensed under an MIT and a GNU General Public License (GPL) license. It's free, yo, in multiple ways!
- It's small (18 KB minified) and gzipped (114 KB, uncompressed).
- It's incredibly popular, which is to say it has a large community of users and a healthy amount of contributors who participate as developers and evangelists.
- It normalizes the differences between web browsers so that you don't have to.
- It's intentionally a lightweight footprint with simple yet clever plugin architecture.
- Its repository of plugins is vast and has seen steady growth since jQuery's release.
- Its API is fully documented, including inline code examples, which in the world of JavaScript libraries is a luxury. Heck, any documentation at all was a luxury for years.

- It's friendly, which is to say it provides helpful ways to avoid conflicts with other JavaScript libraries.
- Its community support is actually fairly useful, including several mailing lists, IRC channels, and a freakishly insane amount of tutorials, articles, and blog posts from the jQuery community.
- It's openly developed, which means anyone can contribute bug fixes, enhancements, and development help.
- Its development is steady and consistent, which is to say the development team is not afraid of releasing updates.
- Its adoption by large organizations has and will continue to breed longevity and stability (e.g., Microsoft, Dell, Bank of America, Digg, CBS, Netflix).
- It's incorporating specifications from the W3C before the browsers do. As an example, jQuery supports a good majority of the CSS3 selectors.
- It's currently tested and optimized for development on modern browsers (Chrome 1, Chrome Nightly, IE 6, IE 7, IE 8, Opera 9.6, Safari 3.2, WebKit Nightly, Firefox 2, Firefox 3, Firefox Nightly).
- It's downright powerful in the hands of designer types as well as programmers. jQuery does not discriminate.
- Its elegance, methodologies, and philosophy of changing the way JavaScript is written is becoming a standard in and of itself. Consider just how many other solutions have borrowed the selector and chaining patterns.
- Its unexplainable by-product of feel-good programming is contagious and certainly unavoidable; even the critics seem to fall in love with aspects of jQuery.
- Its documentation has many outlets (e.g., API browser, dashboard apps, cheat sheets) including an offline API browser (AIR application).
- It's purposely bent to facilitate unobtrusive JavaScript practices.
- It has remained a JavaScript library (as opposed to a framework) at heart while at the same time providing a sister project for user interface widgets and application development (jQuery UI).

- Its learning curve is approachable because it builds upon concepts that most developers and designers already understand (e.g., CSS and HTML).

11.2 Adding jQuery to Web Pages

There are several method to add jQuery to web page, some of the common way to add jQuery are:

(i) Local Installation

(ii) CDN Based Version

(i) Local Installation:-

You can download jQuery from its official website (Link: https://jquery.com/)

The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

(ii) CDN Based Version:-

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network). Both Google and Microsoft host jQuery. To use jQuery from Google or Microsoft, use one of the following:

Google CDN:

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
</head>
```

Microsoft CDN:

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.1.min.js"></script>
</head>
```

11.3 jQuery Syntax and Selector

jQuery Syntax is very simple and easy to use. We will try to understand the basic syntax of jQuery by following the steps:

1. Frist we have to add the jQuery onto our web page as discussed in section 11.2
2. Call the jQuery Library functions as

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

   All the stuff like reads or manipulates the DOM are inside $(document).ready() function. This will prevent any jQuery code from running before the document is finished loading (is ready). It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section. It possible that some methods are fail to execute if we try to run the methods before the document is fully loaded.
3. Now inside $(document).ready() function, we have some stuff related to read or manipulate DOM. The best way is using "selector". jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. This is same as we done in CSS. The syntax of selector is start with the dollar sign and parentheses like $().

   The following table will show different type of selector to use:

S.No.	Selector with description	Example
1.	Name: Selects all elements which match with the given element Name.	• $('p'): Selects all elements with a tag name of p in the document. • $('div'): Selects all elements

		with a tag name of div in the document.
2.	#ID: Selects a single element which matches with the given ID.	• $('#myid') – Selects a single element with the given id myid. • $('div#yourid') – Selects a single division with the given id yourid.
3.	.Class: Selects all elements which matches with the given Class.	• $('.big') – Selects all the elements with the given class ID big. • $('p.small') – Selects all the paragraphs with the given class ID small. • $('.big.small') – Selects all the elements with a class of big and small.
4.	Universal (*): Selects all elements available in a DOM.	• $('*') selects all the elements available in the document.
5.	Multiple Elements E, F, G: Selects the combined results of all the specified selectors E, F or G.	• $('div, p') – selects all the elements matched by div or p. • $('p strong, .myclass') – selects all elements matched by strong that are descendants of an element matched by p as well as all elements that have a class of myclass. • $('p strong, #myid') – selects

		a single elements matched by strong that is descendant of an element matched by p as well as element whose id is myid.

4. After defining the selector, we have to specify the action on the selected selector. Basic syntax is:

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

Where,

- $ sign to define/access jQuery
- (selector) to "query (or find)" HTML elements
- jQuery action() to be performed on the element(s)

Example:

Example	Description
$(this).hide()	hides the current element.
$("p").hide()	hides all <p> elements.
$(".test").hide()	hides all elements with class="test".
$("#test").hide()	hides the element with id="test".

The following example will hide all the contents inside 'p' element.

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>

<script>
$(document).ready(function(){
   $("p").hide();
});
</script>
</head>
<body>

<h2>This is a heading</h2>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

11.4 jQuery Event Methods and their syntax

Events are response of the action taken by web user while visiting the web page. For example: user click on button, moving a mouse over an element, etc.

```
$("p").click(function(){
 // action goes here!!
});
```

The above code is for click event, when user clicks on the p element then the function inside click will executed.

Suppose we want to hide 'p' element then we will write as

```
$("p").click(function(){
   $(this).hide();
});
```

The following are some common jQuery Event method

1. **Mouse Event**

S.No.	Event Method	Example	Description
1.	click	$("p").click(function(){ $(this).hide(); });	Hide current element if user clicks on 'p' element.

2.	dblclick	$("p").dblclick
		(function(){
		        $(this).hide();
		});
			Hide current element if user double click on 'p' element.

3.	mouseenter	$("p"). mouseenter
		(function(){
		        $(this).hide();
		});
			Hide current element if user enter the mouse pointer on 'p' element.

4.	mouseleave	$("p"). mouseleave
		(function(){
		        $(this).hide();
		});
			Hide current element if user's mouse pointer leaves 'p' element.

2. **Keyboard Events**

S.No.	Event Method	Example	Description

1.	keypress	$(" input "). keypress
		(function(){
		        $(this).hide();
		});
			Hide current element if user clicks on 'p' element.

2.	keydown	$("p"). keydown
		(function(){
		        $(this).hide();
		});
			Hide current element if user double click on 'p' element.

3.	keyup	$("p"). keyup (function(){	Hide current element if user enter the mouse pointer on

		$(this).hide(); });	'p' element.

3. **Form Events**

S.No.	Event Method	Example	Description

1.	submit	$(" input "). submit (function(){
		alert('Conformed');
		});
			The function is executed when the form submitted.

2.	change	$("input").change(function(){ $(this).css("background-color", "green"); });	The function is executed when the form field gets changes.
3.	focus	$("input").focus(function(){ $(this).css("background-color", "blue"); });	The function is executed when the form field gets focus.
4.	blur	$("input").blur(function(){ $(this).css("background-color", "#ffffff"); });	The function is executed when the form field loses focus

4. **Document/Window Events**

S.No.	Event Method	Example	Description

1.	load	$("img").load(function(){
		alert("Image loaded.");
		});
			The load event occurs when a specified element has been loaded.

2.	resize	$(window).resize(function() { $("#log").append("<div>Handler for	The resize event is sent to the window element when the size of the browser window

		.resize() called.</div>");
		});
			changes

3.	scroll	$(window
		).scroll(function() {
		  $("#log").append(
		"<div>Scroll
		Happen.</div>");
		});
			The function is executed when the scroll the window.

4.	unload	$(window
		).unload(function() {
		  return "Bye now!";
		});
			This will display an alert when a page is unloaded.

11.5 jQuery Effects

jQuery comes with a number of effects and the robust low-level animation method for creating your own custom effects.

- The effects can do the following to the DOM:
- Hiding and showing elements in a toggle fashion
- Scaling and simultaneously fading elements in and out of view
- Sliding up and down and toggling
- Fading in and out and to a specific opacity

Some of the jQuery effects methods are:

(i) hide(), show() and toggle() method

(ii) Fading effects method: fadeIn(), fadeOut(), fadeToggle(), and fadeTo()

(iii) Sliding effects method: slideDown(),slideUp(), and slideToggle()

(iv) Animation method: animate()

(v) Other method: stop()

Callback Functions

jQuery Method Chaining

(i) ***hide(), show() and toogle() method***: These are used for showing, hiding and toggling between hide and show the elements.

Syntax of show() method:

```
[selector].show( speed, [callback] );
```

Where,

speed: A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

callback: This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

Syntax of hide() method:

```
[selector].hide( speed, [callback] );
```

The explanation of the parameters are same as of show() method.

Example: The following code will hide div when clicking on hide button and show it again when we click on show button.

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js">
</script>
<script type="text/javascript" language="javascript">
$(document).ready(function() {
$("#show").click(function () {
$(".mydiv").show( 1000 );
});
$("#hide").click(function () {
$(".mydiv").hide( 1000 );
});
});
</script>
</head>
<body>
<div class="mydiv">
This is a SQUARE
</div>
<input id="hide" type="button" value="Hide" />
<input id="show" type="button" value="Show" />
</body></html>
```

You can toggle between the hide() and show() methods with the toggle() method.

Example:

```
$("button").click(function(){
   $("p").toggle();
});
```

(ii) ***Fading effects method:*** With the help of these methods, you can fade the elements in and out of visibility. These include the following methods:

- **fadeIn() method** : It is use to fade the element In of visibility in a hidden element.

  Syntax:

  ```
  $(selector).fadeIn(speed,callback);
  ```

  Example:

  ```
  $("button").click(function(){
     $("#div1").fadeIn();
            $("#div2").fadeIn("slow");
           $("#div3").fadeIn(3000);
  });
  ```

- **fadeOut()method**: It is used to fade the element Out of visibility in a visible element.

  Syntax:

  ```
  $(selector).fadeOut(speed,callback);
  ```

  Example:

  ```
  $("button").click(function(){
     $("#div1").fadeOut();
     $("#div2").fadeOut("slow");
     $("#div3").fadeOut(3000);
  });
  ```

- **fadeToggle()method**: It is used to toggle between fade in and fade out.

  Syntax:

  ```
  $(selector).fadeToggle(speed,callback);
  ```

  Example:

  ```
  $("button").click(function(){
     $("#div1").fadeToggle();
     $("#div2").fadeToggle("slow");
     $("#div3").fadeToggle(3000);
  });
  ```

- **fadeTo()method**: It is used to fading the element upto certain value of opacity (in between 0 to 1).

  Syntax:

  ```
  $(selector).fadeTo(speed,opacity,callback);
  ```

  Example:

  ```
  $("button").click(function(){
     $("#div1").fadeTo("slow", 0.15);
     $("#div2").fadeTo("slow", 0.4);
     $("#div3").fadeTo("slow", 0.7);
  ```

```
});
```

(iii) ***Sliding effects method:*** These methods are used to slide elements up and down.

- **slideDown() method:** It is used to slide down the elements.

  Syntax:

  ```
  $(selector).slideDown(speed,callback);
  ```

  Example:

  ```
  $("#flip").click(function(){
     $("#panel").slideDown();
  });
  ```

- **slideUp() method:** It is used to slide up the elements.

  Syntax:

  ```
  $(selector).slideUp(speed,callback);
  ```

  Example:

  ```
  $("#flip").click(function(){
     $("#panel").slideUp();
  });
  ```

- **slideToogle() method:** This method is used to toggle the sliding effect between slide Up and Down.

  Syntax:

  ```
  $(selector).slideToggle(speed,callback);
  ```

  Example:

  ```
  $("#flip").click(function(){
     $("#panel").slideToggle();
  });
  ```

(iv) ***Animation method:*** This include animate() method, which is used to create custom animation.

Syntax:

```
$(selector).animate({params},speed,callback);
```

The required params parameter defines the CSS properties to be animated.

Example:

```
$("button").click(function(){
$("div").animate({left: '250px'});
});
```

The above example will moves a <div> element to the right, until it has reached a left property of 250px.

(v) ***jQuery Stop Animations:*** This include stop() method, which is used to stop the animation before it is finished. It works for all jQuery effect functions, including sliding, fading and custom animations.

Syntax:

```
$(selector).stop(stopAll,goToEnd);
```

The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.

The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.

So, by default, the stop() method kills the current animation being performed on the selected element.

Example:

```
$("#stop").click(function(){    $("#panel").stop();});
```

The above statement will stop all the animation applied on element whose id is panel.

11.6 jQuery HTML

In this section, we will study some powerful methods for changing and manipulating HTML elements and their attributes. You 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. For manipulating DOM, jQuery has some simple methods like text(), html() and val(). These methods can be used for retrieving information from DOM elements for later use.

The DOM manipulation methods can be categorized as

(i) ***Methods used for getting/setting contents of elements***: These methods are used for exacting information from elements or setting the information into the elements. These includes:

- **text() method**- By using this method, we can get the combined text contents of all matched elements as

  Syntax:

  ```
  selector.text( );
  ```

  Example:

  ```
  $(document).ready(function() {
  var content = $(".p").text();
  alert (content);
  });
  ```

  We can also set the text content of all matched elements as

  Syntax:

  ```
  selector.text( val);
  ```

  Where,

  val is any string.

  Example:

  ```
  $(document).ready(function() {
  var content = $(".p1").text();
  $(".p2").text(content);
  });
  ```

- html() method: The html() method gets the html contents of the first matched element. This property is not available on XML documents but it works for XHTML documents.

  Syntax:

  ```
  selector.html( );
  ```

  Example:

  ```
  $(document).ready(function() {
  ```

```
var content = $("p").html();
$("div").html( content );
});
```

- val() method: It is used to get the input value of the first matched element.

  Syntax:

  ```
  selector.val( );
  ```

  Example:

  ```
  $(document).ready(function() {
  var content = $("input").val();
  $(".p1").text(content);
  });
  ```

  For setting the input value of every matched element, we use val(val) method:

  Syntax:

  ```
  selector.val( val )
  ```

  Example:

  ```
  $(document).ready(function() {
  var content = $("input").val();
  $("input").val( content );
  });
  ```

  NOTE: If this method is called on radio buttons, checkboxes, or select options then it would check, or select them at the passed value.

- **attr() method**: This method can be used to set a key/value object as properties to all matched elements.

  Syntax:

  ```
  selector.attr({property1:value1, property2:value2})
  ```

  Here is the description of all the parameters used by this method:

  property: This is the CSS property of the matched element.

  value: This is the value of the property to be set.

Example:

```
$(document).ready(function() {
$("img").attr({
src: "/images/jquery.jpg",
title: "jQuery",
alt: "jQuery Logo"
});
});
```

This method sets a single property to a computed value, on all matched elements.

Syntax:

```
selector.attr( key, func);
```

Where,

Key is the name of the property to set.

Func is a function returning the value to set. This function would have one argument which is index of current element.

Example:

(ii) ***Methods used for adding the elements onto DOM:*** These includes append(), prepend(), after(), before().

append() method: This method is used to append the content to the inside of all matched elements.

Syntax:

selector.append(content);

Where,

content is the text content to be append inside the selector.

Example: When user click on div text 'Hello' is append to the current selector.

```
$(document).ready(function() {

$("div").click(function () {
```

```
$(this).append('Hello' );
});
});
```

prepend() method: This method is used to prepend the content to the inside of all matched elements.

Syntax:

selector.prepend(content);

Where,

content is the text content to be prepend inside the selector.

Example: When user click on div the text ‘Hello’ will be prepend to the current selector.

```
$(document).ready(function() {
$("div").click(function () {
$(this).prepend('Hello' );
});
});
```

after() method: This method inserts content after each of the matched elements.

Syntax:

```
selector.after(content);
```

Example:

```
$(document).ready(function() {
```

```
$("div").click(function () {

$(this).after('<div class="div"></div>' );

});

});
```

before() method: This method inserts content before each of the matched elements.

Syntax:

```
selector.before (content);
```

Example:

```
$(document).ready(function() {

$("div").click(function () {

$(this).before('<div class="div"></div>' );

});

});
```

(iii) ***Methods used for removing the elements onto DOM:*** These includes remove() and empty() method.

remove() method: The remove(expr) method removes all matched elements from the DOM. This does NOT remove them from the jQuery object, allowing you to use the matched elements further.
Syntax:

```
selector.remove( expr )
```

Where,

expr: This is an optional jQuery expression to filter the set of elements to be removed

empty() method: It removes all child nodes from the set of matched elements.

Syntax:

```
selector.empty( );
```

(iv) ***Methods used for CSS Classes***: addClass(), removeClass(), toogleClass() and css()

addClass() method

removeClass() method

toogleClass() method

css() method

(v) jQuery Dimension:

11.7 jQuery AJAX

As we studied in Unit-4, AJAX is an acronym stands for Asynchronous JavaScript and XML. It is used to load data from server without refreshing the web page. In this section, we will explain the rich set of AJAX methods provided by jQuery to build next generation web application. Some of the methods are:

load() method: This method is use to load any static and dynamic data using jQuery AJAX.

Syntax:

```
[selector].load( URL, [data], [callback] );
```

URL: The URL of the server-side resource to which the request is sent. It could be a CGI, ASP, JSP, or PHP script which generates data dynamically or out of a database.

data: This optional parameter represents an object whose properties are serialized into properly encoded parameters to be passed to the request. If specified, the request is made using the POST method. If omitted, the GET method is used.

callback: A callback function invoked after the response data has been loaded into the elements of the matched set. The first parameter passed to this function is the response text received from the server and second parameter is the status code.

Example: The following example will load the text file when user clicks on button.

```
<html>
<head>
<title>the title</title>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"> </script>
<script type="text/javascript" language="javascript">
$(document).ready(function(){
   $("#driver").click(function(){
      $("#stage").load("demo.txt");
   });
});
</script>
</head>
```

```
<body>
<p>Click on the button to load result.html file:</p>
<div id="stage" style="background-color:yellow;">
STAGE
</div>
<input type="button" id="driver" value="Load Data" />
</body>
</html>
```

When user click on the button the output is (i.e. the content of demo.txt):

```
jQuery and AJAX is FUN!
This is some text in a paragraph.
```

$.get() and $.post() Method: These methods are used to request data from the server with an HTTP GET or POST request.

$.get() method is used to load data from the server using a GET HTTP request. The method returns XMLHttpRequest object.

Syntax:

```
$.get( url, [data], [callback], [type] )
```

url: A string containing the URL to which the request is sent

data: This optional parameter represents key/value pairs that will be sent to the server.

callback: This optional parameter represents a function to be executed whenever the data is loaded successfully.

type: This optional parameter represents type of data to be returned to callback

function: "xml", "html", "script", "json", "jsonp", or "text".

Example: In this example, when we click on the button the HTTP get request is send and return the data, the data will be shown on the element whose id is “div1”.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("demo.php", function(data, status){
      $("#div1").html(data);
    });
  });
});
</script>
</head>
<body>
<div id="div1" style="background-color:yellow;">Hello</div>
<button>Send an HTTP GET request to a page and get the result back</button>
</body>
</html>
```

demo.php contain the following code:

```
<?php
echo “hello this is server response with jQuery and AJAX”;
?>
```

$.post() method is used to request data from the server using an HTTP POST request. This method can also be used to get some data from the server. However, the $.post() method NEVER caches data, and is often used to send data along with the request. However, $.get() returns cached data.

Syntax:

```
$.post( url, [data], [callback], [type] )
```

url: A string containing the URL to which the request is sent

data: This optional parameter represents key/value pairs or the return value of the .serialize() function that will be sent to the server.

callback: This optional parameter represents a function to be executed whenever the data is loaded successfully.

type: This optional parameter represents a type of data to be returned to callback function: "xml", "html", "script", "json", "jsonp", or "text".

Example: In the example, name "Neeraj Arora" is send to server, the server retrieve the send data and response the data "Welcome Neeraj Arora" using php.

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
   $("button").click(function(){
      $.post("demo.php",
      {
        name: "Neeraj Arora",
       },
      function(data,status){
         $("#div1").html(data);
      });
   });
});
</script>
</head>
<body>
```

```
<div id="div1" style="background-color: yellow;">Hello</div>
<button>Send an HTTP POST request to a page and get the result back</button>
</body>
</html>
```

demo.php contains the following code:

```
<?php
if( $_REQUEST["name"] )
{
$name = $_REQUEST['name'];
echo "Welcome ". $name;
}
?>
```

11.8 Self Learning Exercise

Q.1 jQuery is/are:

a) Open Source
b) JavaScript Library
c) Support cross browser development
d) All of the above

Q. 2 AJAX stands for:

a) All Java All XML.
b) All JavaScript Ajax XML,
c) Asynchronous JavaScript and XML.
d) None of the above.

Q.3 What is the use of **$(this).hide()**?

a) hides the current element.
b) hides all elements.
c) hides the element with id="this".
d) All of the above.

11.9 Summary

- jQuery is an open source JavaScript library that simplifies the interactions between an HTML document and javaScript
- jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.
- Events in jQuery are response of the action taken by web user while visiting the web page.
- For manipulating DOM, jQuery has some simple methods like text(), html() and val().
- Asynchronous JavaScript and XML is used to load data from server without refreshing the web page

11.10 Answers to Self-Learning Exercise

Q.1 (d)

Q.2 (c)

Q.3 (a)

11.11 Exercise

Q.1 Explain all the features of jQuery.

Q.2 How many methods are there to add jQuery? Explain each method in detail.

Q.3 What is the use of event in jQuery? Explain mouse event and key board event with one example.

Q.4 How jQuery methods are used for changing and manipulating HTML elements and their attributes? Explain in detail.

References and Suggested Readings

1. Simon St.Laurent, jQuery Cookbook, O'Reilly Media, Inc.,November 2009, Frist Edition.

2. Papazoglou, Web Services: Principles and Technology (2nd edition); ISBN: 978-0-273-73216-7, Prentice Hall, 2012.
3. Alonso, Casati, Kuno and Machiraju, Web Services: Concepts, Architectures and Applications; ISBN: 3540440089, Springer, 2004.
4. Cerami, Web Services Essentials; ISBN: 0596002246, O'Reilly, 2002.
5. Seomoz Analytics Tool. (2012). Retrieved December 25, 2012 from http://www.seomoz.org.

UNIT-12
Search Engines

Structure of the Unit

12.0 Objective

In this chapter we shall focus upon the following topics

- Define search engine & how does it work
- Keywords and Metadata sculpting
- Search Engine Development & Optimization
- SEO Web design
- Effective Content Writing Plan
- Achieving high ranking

- SEO analysis intervals

12.1 Define Search Engine & How does it work

A web search engine is a tool that allows user to find the information on the World Wide Web. It's possible to think of the internet as the world's biggest library - but instead of books, its shelves contain billions of individual web pages. Imagine being in such a vast library. It would take forever to find what you were looking for!

Every library has an index to help you track down the book you want. The internet has something similar in the form of 'search engines'. If Megan wants to find out information on Pest control services she will probably search in Google by typing in:

Pest Control Services
As you know, the World Wide Web is so huge that it is absolutely impossible to find all web pages that reference pest control services. In any case, Megan is not interested in web pages that just mention pest control. Without a search engine it would have been impossible for Megan to find what she wanted.
But we are able to find what we want within seconds using Google!

Search engines come in a number of configurations that reflect the applications they are designed for. Web search engines, such as Google and Yahoo!, must be able to capture, or crawl, many terabytes of data, and then provide sub second response times to millions of queries submitted every day from around the world.

Enterprise search engines—for example, Autonomy8 must be able to process the large variety of information sources in a company and use company-specific knowledge as part of search and related tasks, such as data mining. Data mining refers to the automatic discovery of interesting structure in data and includes techniques such as clustering.

Desktop search engines, such as the Microsoft Vista™ search feature, must be able to rapidly incorporate new documents, web pages, and email as the person creates or looks at them, as well as provide an intuitive interface for searching this very heterogeneous mix of information. There is overlap between these categories with systems such as Google, for example, which is available in configurations for enterprise and desktop search.

Open source search engines are another important class of systems that have different design goals than the commercial search engines. There are a number of these systems, and the Wikipedia page for information retrieval provides links to many of them. Three systems of particular interest are Lucene, Lemur and the system provided with this book, Galago. Lucene is a popular Java-based search engine that has been used for a wide range of commercial applications. The information retrieval techniques that it uses are relatively simple.

Practical issues that impact search engine design also occur for specific applications. The best example of this is spam in web search. Spam is generally thought of as unwanted email, but more generally it could be defined as misleading, inappropriate, or non-relevant information in a document that is designed for some commercial benefit. There are many kinds of spam, but one type that search engines must deal with is spam words put into a document to cause it to be retrieved in response to popular queries. The practice of "spamdexing" can significantly degrade the quality of a search engine's ranking, and web search engine designers have to develop techniques to identify the spam and remove those documents. Figure 12.1 denotes the major issues involved in search engine design.

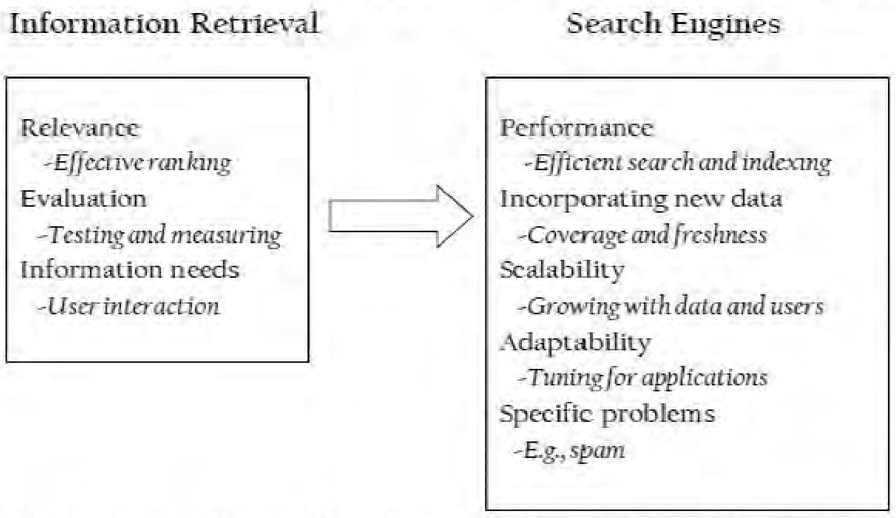


Figure 12.1 Design of Search Engine and Core information retrieval issue

How does it work?

Assume that you are reading a book and want to find reference to a specific word in the book. What do you do? You turn the pages to the end and look in the index! You would then locate the word in the index, find the page numbers mentioned there and flip to the corresponding pages.
Search Engine also works in a same way.

Search engines are constantly building and updating their index to the World Wide Web. They do this by using "spiders" that "crawl" the web and fetch web pages. Then the words used in these web pages are added to the index along with where the words came from.

When someone searches for "pest control services" the search engine already has a list of web pages that refer to "pest control services". The only thing left to do is to sort the web pages in the order of relevance. This is done based on a number of key factors.
Search engines perform several activities in order to deliver search results.

- **Crawling -** Process of fetching all the web pages linked to a website. This task is performed by a software called a crawler or a spider (or Googlebot, in case of Google).
- **Indexing -** Process of creating index for all the fetched web pages and keeping them into a giant database from where it can later be retrieved. Essentially, the process of indexing is identifying the words and expressions that best describe the page and assigning the page to particular keywords.
- **Processing -** When a search request comes, the search engine processes it, i.e., it compares the search string in the search request with the indexed pages in the database.
- **Calculating Relevancy -** It is likely that more than one page contains the search string, so the search engine starts calculating the relevancy of each of the pages in its index to the search string.
- **Retrieving Results -** The last step in search engine activities is retrieving the best matched results. Basically, it is nothing more than simply displaying them in the browser.

Features of Search Engine

- A true search engine is an automated software program that moves around the Web collecting Web Pages to include in its catalog or database.
- It searches when a user requests information from a search engine; not the entire Web.
- Each search engine has its own catalog or database of collected Web Pages, so you will get different results/hits by using different search engines.

Architecture of Search Engine

Our search engine architecture is used to present high-level descriptions of the important components of the system and the relationships between them. It is not a code-level description, although some of the components do correspond to software modules in the Galago search engine and other systems. An architecture is designed to ensure that a system will satisfy the application requirements or goals. The two primary goals of a search engine are:

- Effectiveness (quality): We want to be able to retrieve the most relevant set of documents possible for a query.
- Efficiency (speed): We want to process queries from users as quickly as possible.

The architecture of a search engine is determined by these two requirements. Because we want an efficient system, search engines employ specialized data structures that are optimized for fast retrieval. Because we want high-quality results, search engines carefully process text and store text statistics that help improve the relevance of results.

Search engine components support two major functions, which are known as the indexing process and the query process. The indexing process builds the structures that enable searching, and the query process uses those structures and a person's query to produce a ranked list of documents. Figure 12-2 shows the high-level "building blocks" of the indexing process. These major components are text acquisition, text transformation, and index creation.

The task of the text acquisition component is to identify and make available the documents that will be searched. Text acquisition will more often require building a collection by crawling or scanning the Web, a corporate intranet, a desktop, or other sources of information. In addition to passing documents to the next component in the indexing process, the text acquisition component creates a document data store, which contains the text and metadata for all the documents.

Metadata is information about a document that is not part of the text content, such the document type (e.g., email or web page), document structure, and other features, such as document length.

The text transformation component transforms documents into index terms or features. Index terms, as the name implies, are the parts of a document that are stored in the index and used in searching. The simplest index term is a word, but not every word may be used for searching. A "feature" is more often used in the field of machine learning that is used to represent its content, which also describes an index term. Examples of other types of index terms or features are phrases, names of people, dates, and links in a web page. Index terms are sometimes simply referred to as "terms." The set of all the terms that are indexed for a document collection is called the index vocabulary.

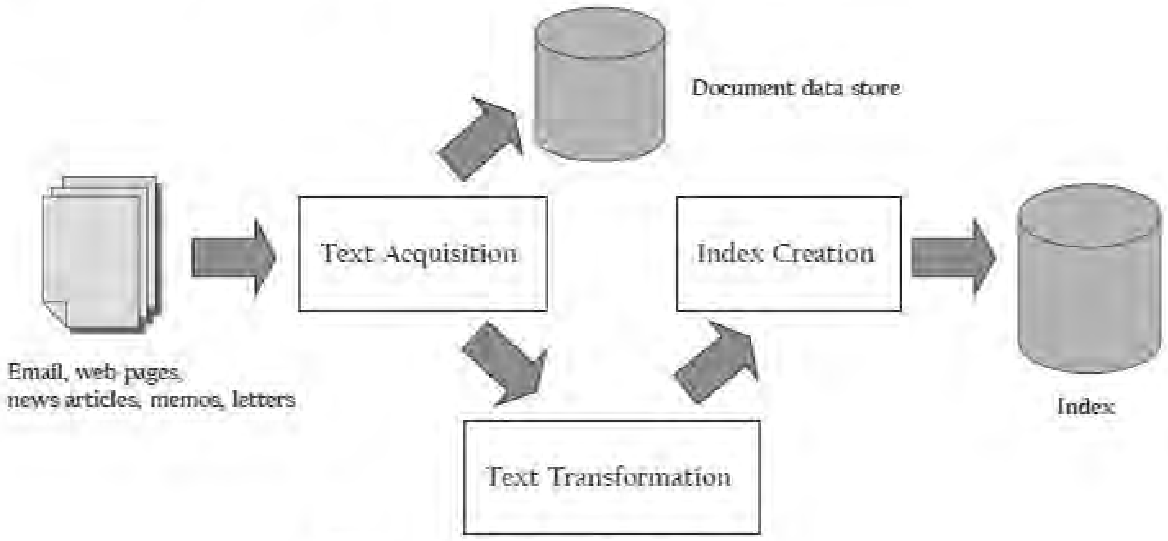


Figure 12.2 the Indexing Process

Figure 12.3 shows the building blocks of the query process. The major components are user interaction, ranking, and evaluation. The user interaction component provides the interface between the person doing the searching and the search engine. One task for this component is accepting the user's query and transforming it into index terms. Another task is to take the ranked list of documents from the search engine and organize it into the results shown to the user. This includes, for example, generating the snippets used to summarize documents. The document data store is one of the sources of information used in generating the results. Finally, this component also provides a range of techniques for refining the query so that it better represents the information need.

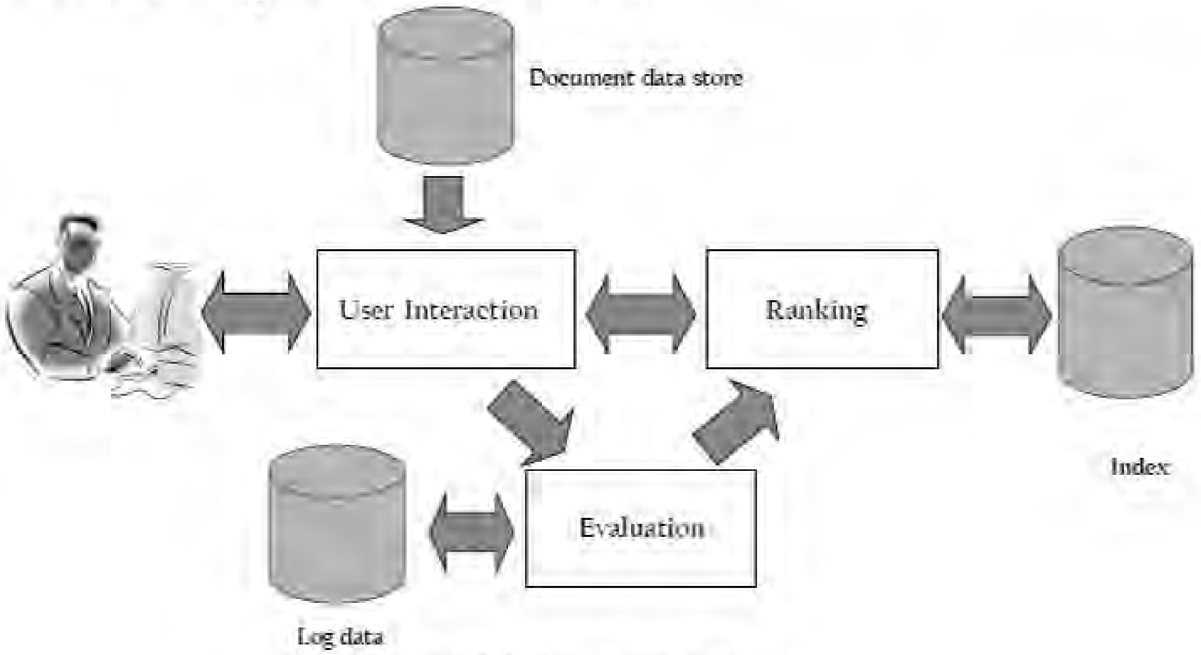


Figure 12.3 the query process

Keywords are what help your site get recognized by Internet search engines, and as a result, help would-be customers find your Internet presence. There are many strategies that can be employed to ensure the likelihood of your website coming up in keyword searches:

- Use a keyword tool to help you find the most targeted keywords for your site.
- Pick themes or keyword baskets around which you can optimize the various sections of your site.
- Targeting keyword phrases is a much better idea than trying to target individual words. Keyword phrases tend to be easier to rank well for AND they typically convert far better than individual words.
- Target different keyword phrases on each page.

- Target no more than one or two primary and two or three secondary keyword phrases per page.
- If you generate hundreds (or even thousands) of pages of content, make sure they read well and have unique content. Over time, if people cite your content, your page will start to rank for many different terms as long as it is unique and targeted around a theme.

Keywords and metadata play a vital role in search engine optimization. In the SEO, the first step is to choose a keyword i.e. done by in following manner:

Read the page's content and identify two keywords that are most relevant to the overall page content. Choose one primary keyword relevant to the page's content and one variation of that keyword (example plural variation or two closely related keywords) per page. If you can't identify one primary keyword for a page, you'll need to create new website pages to separate the different content. If it's not clear to you what page is about, then your visitors and the search engines won't be able to understand the page either.

Page Title

The page title appears as the blue, bolded, underlined text on a Google search results page, and also on the top left the browser bar. The page title should follow these guidelines:

- Be under 70 characters with no more than two long-tail keywords per page title.
- The primary keyword should appear first.
- Each keyword phrase should be separated by pipes (|).
- Each page title on your website should be unique.
- Except for your homepage and contact us page, each page title should NOT include your business name.

Meta Description

The Meta description appears on a Google search results page under the Page Title. The Meta description helps people decide whether to click on your result, or a result above or below you. Think of it as a call to action. The Meta description should follow these guidelines:

- Be under 150 characters (but not under 100 characters; take advantage of the space you have).
- Incorporate the primary keyword and at least one secondary keyword.
- Provide a valuable, compelling reason for why someone should visit the page.
- Include keywords in a conversational format; don't just cram in keywords for the sake of listing them.

12.3 Search Engine Development & Optimization

SEO is short for search engine optimization. Search engine optimization is a methodology of strategies, techniques and tactics used to increase the amount of visitors to a website by obtaining a high-ranking placement in the search results page of a search engine (SERP) - including Google, Bing, Yahoo and other search engines.

What is SEO Copywriting?

SEO copywriting is the technique of writing viewable text on a web page in such a way that it reads well for the surfer, and also targets specific search terms. Its purpose is to rank highly in the search engines for the targeted search terms. Along with viewable text, SEO copywriting usually optimizes other on-page elements for the targeted search terms. These include the Title, Description, Keywords tags, headings, and alternative text.

The idea behind SEO copywriting is that search engines want genuine content pages and not additional pages often called "doorway pages" that are created for the sole purpose of achieving high rankings.

On-Page and Off-Page SEO

Conceptually, there are two ways of optimization:

- **On-Page SEO** – It includes providing good content, good keywords selection, putting keywords on correct places, giving appropriate title to every page, etc.
- **Off-Page SEO** - It includes link building, increasing link popularity by submitting open directories, search engines, link exchange, etc.

SEO techniques are classified into two broad categories:

White Hat SEO – Techniques that search engines recommend as part of a good design.

Black Hat SEO – Techniques that search engines do not approve and attempt to minimize the effect of. These techniques are also known as spamdexing.

White Hat SEO

An SEO tactic is considered as White Hat if it has the following features:
It conforms to the search engine's guidelines.

- It does not involve in any deception.
- It ensures that the content a search engine indexes, and subsequently ranks, is the same content a user will see.
- It ensures that a web page content should have been created for the users and not just for the search engines.
- It ensures good quality of the web pages.
- It ensures availability of useful content on the web pages.

Always follow a White Hat SEO tactic and do not try to fool your site visitors. Be honest and you will definitely get something more.

Black Hat or Spamdexing

An SEO tactic is considered as Black Hat or Spamdexing if it has the following features:

- Attempting ranking improvements that are disapproved by the search engines and/or involve deception.

- Redirecting users from a page that is built for search engines to one that is more human friendly.
- Redirecting users to a page that was different from the page the search engine ranked.
- Serving one version of a page to search engine spiders/bots and another version to human visitors. This is called Cloaking SEO tactic.
- Using hidden or invisible text or with the page background color, using a tiny font size or hiding them within the HTML code such as "no frame" sections.
- Repeating keywords in the metatags, and using keywords that are unrelated to the website content. This is called metatag stuffing.
- Calculated placement of keywords within a page to raise the keyword count, variety, and density of the page. This is called keyword stuffing.
- Creating low-quality web pages that contain very little content but are instead stuffed with very similar keywords and phrases. These pages are called Doorway or Gateway Pages.
- Mirror websites by hosting multiple websites – all with conceptually similar content but using different URLs.
- Creating a rogue copy of a popular website which shows contents similar to the original to a web crawler, but redirects web surfers to unrelated or malicious websites. This is called page hijacking.

Always stay away from any of the above Black Hat tactics to improve the rank of your site. Search engines are smart enough to identify all the above properties of your site and ultimately you are not going to get anything.

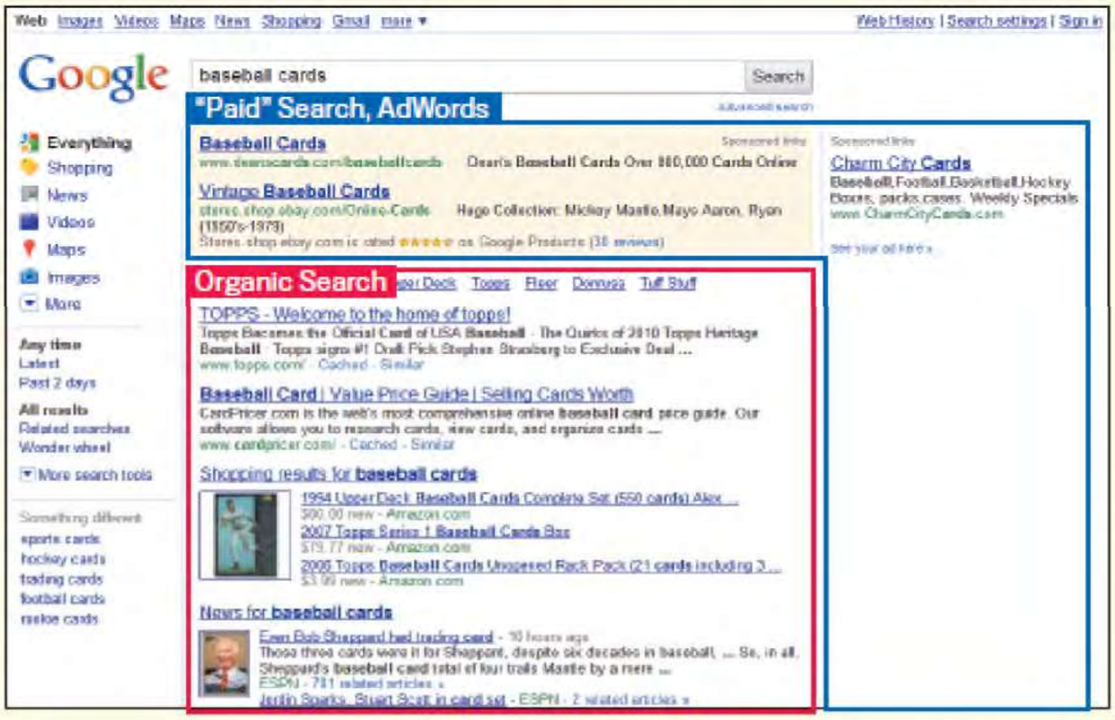
12.4 SEO Web Design

SEO is an important technique for designing efficient and optimized web site. To design an effective web pages we should follow various parameters. Which are as follows:

An example may help our explanations, so we've created a fictitious website to follow throughout the chapter. Here's some background information about the site we'll use:

“Website/business name: "Brandon's Baseball Cards"
Domain name: brandonsbaseballcards.com
Focus: Online-only baseball card sales, price guides, articles, and news content
Size: Small, ~250 pages”

Search engine optimization affects only organic search results, not paid or



baseball cards

Sponsored links

Baseball Cards
www.deanscards.com/baseballcards Dean's Baseball Cards Over 800,000 Cards Online

Vintage Baseball Cards
stores.shop.ebay.com/Online-Cards Huge Collection: Mickey Mantle, Mays Aaron, Ryan (1950's-1979)
Stores.shop.ebay.com is rated ★★★★★ on Google Products (38 reviews)

Sponsored links

Charm City Cards
Baseball, Football, Basketball, Hockey
Boxes, packs, cases. Weekly Specials
www.CharmCityCards.com

See your ad here »

Upper Deck Topps Fleer Donruss Tuff Stuff

TOPPS - Welcome to the home of topps!
Topps Becomes the Official Card of USA **Baseball** - The Quirks of 2010 Topps Heritage **Baseball** - Topps signs #1 Draft Pick Stephen Strasburg to Exclusive Deal ...
www.topps.com/ - Cached - Similar

Baseball Card | Value Price Guide | Selling Cards Worth
CardPricer.com is the web's most comprehensive online **baseball card** price guide. Our software allows you to research cards, view cards, and organize cards ...
www.cardpricer.com/ - Cached - Similar

Shopping results for **baseball cards**
1994 Upper Deck Baseball Cards Complete Set (550 cards) Alex ...
$90.00 new - Amazon.com
2007 Topps Series 1 Baseball Cards Box
$79.77 new - Amazon.com
2008 Topps Baseball Cards Unopened Rack Pack (21 cards including 3 ...
$3.99 new - Amazon.com

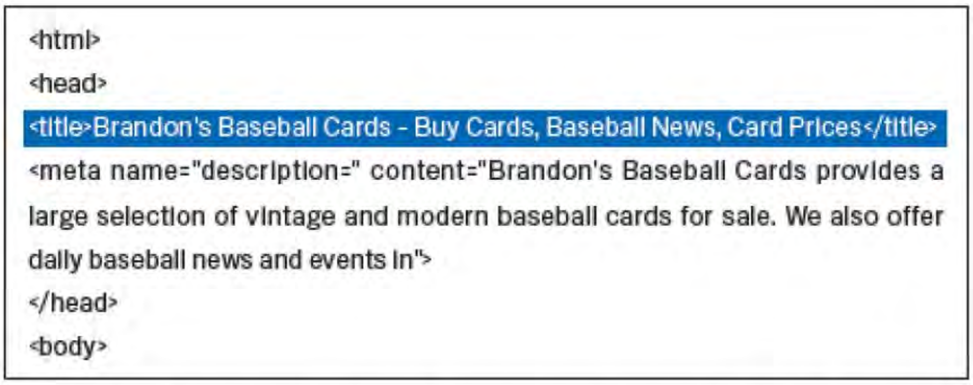
News for **baseball cards**
Even Bob Sheppard had trading card - 10 hours ago
Those three cards were it for Sheppard, despite six decades in baseball. ... So, in all, Sheppard's **baseball card** total of four trails Mantle by a mere ...
ESPN - 781 related articles »
Jerlin Sparks, Stuart Scott in card set - ESPN - 2 related articles »

"sponsored" results such as Google AdWords.

Figure 12.4 “Paid” Search, AdWords and Organic Search

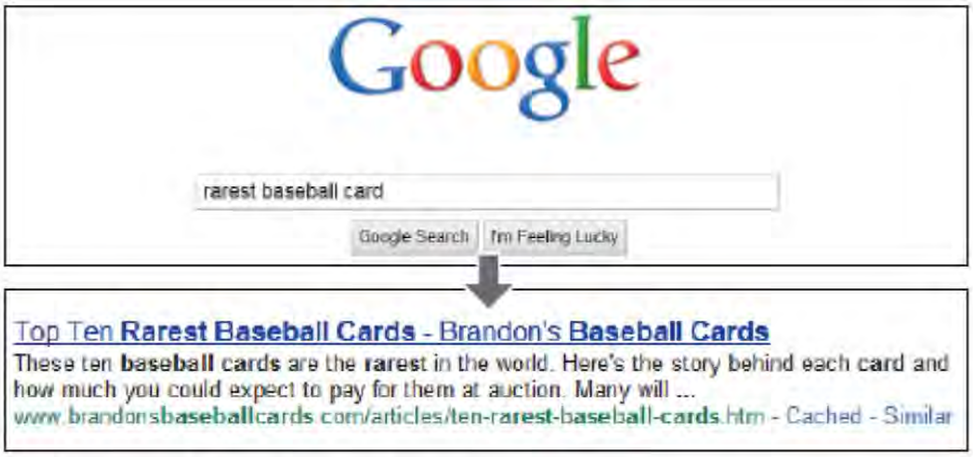
1. **Create Unique and Accurate page title**

   A title tag tells both users and search engines what the topic of a particular page is. The <title> tag should be placed within the <head> tag of the HTML document. Ideally, you should create a unique title for each page on your site.



```
<html>
<head>
<title>Brandon's Baseball Cards - Buy Cards, Baseball News, Card Prices</title>
<meta name="description=" content="Brandon's Baseball Cards provides a
large selection of vintage and modern baseball cards for sale. We also offer
daily baseball news and events in">
</head>
<body>
```

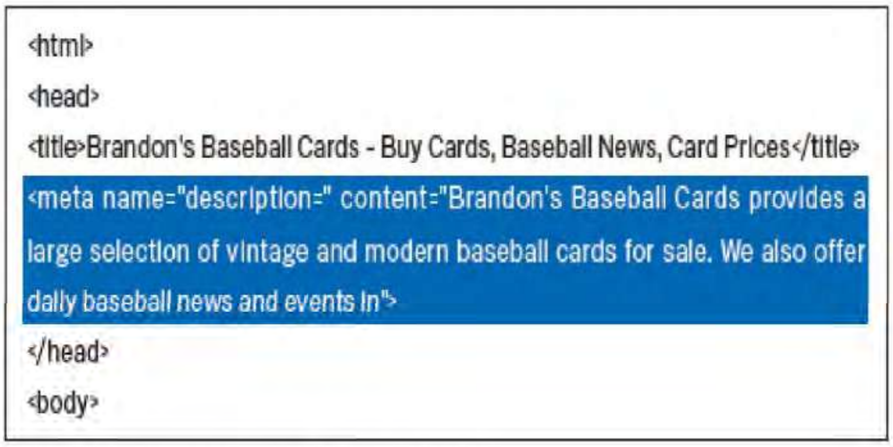
The title for your homepage can list the name of your website/ business and could include other bits of important information like the physical location of the business or maybe a few of its main focuses or offerings.



Top Ten **Rarest Baseball Cards** - Brandon's **Baseball Cards**
These ten **baseball cards** are the **rarest** in the world. Here's the story behind each **card** and how much you could expect to pay for them at auction. Many will ...
www.brandonsbaseballcards.com/articles/ten-rarest-baseball-cards.htm - Cached - Similar

2. Make use of the "description" meta tag

A page's description Meta tag gives Google and other search engines a summary of what the page is about. Whereas a page's title may be a few words or a phrase, a page's description Meta tag might be a sentence or two or a short paragraph. Google Webmaster Tools provides a handy content analysis section that'll tell you about any description meta tags that are either too short, long, or duplicated too many times (the same information is also shown for <title> tags). Like the <title> tag, the description Meta tag is placed within the <head> tag of your HTML document.

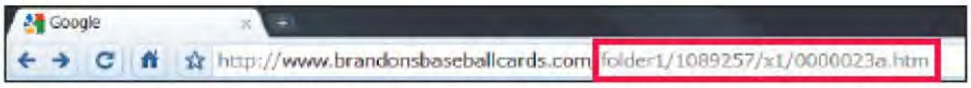


```
<html>
<head>
<title>Brandon's Baseball Cards - Buy Cards, Baseball News, Card Prices</title>
<meta name="description=" content="Brandon's Baseball Cards provides a
large selection of vintage and modern baseball cards for sale. We also offer
daily baseball news and events in">
</head>
<body>
```

3. **Improve the structure of your URL**

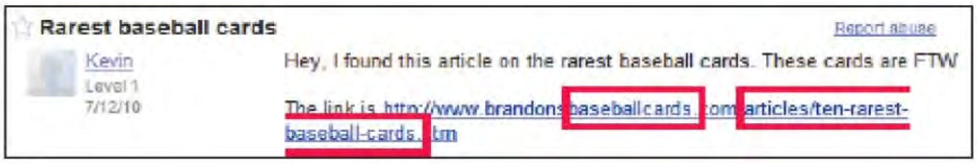
Creating descriptive categories and filenames for the documents on your website can not only help you keep your site better organized, but it could also lead to better crawling of your documents by search engines. Also, it can create easier, "friendlier" URLs for those that want to link to your content. Visitors may be intimidated by extremely long and cryptic URLs that contain few recognizable words.

URLs like (1) can be



http://www.brandonsbaseballcards.com/folder1/1089257/x1/0000023a.htm

(1) A URL to a page on our baseball card site that a user might have a hard time with.



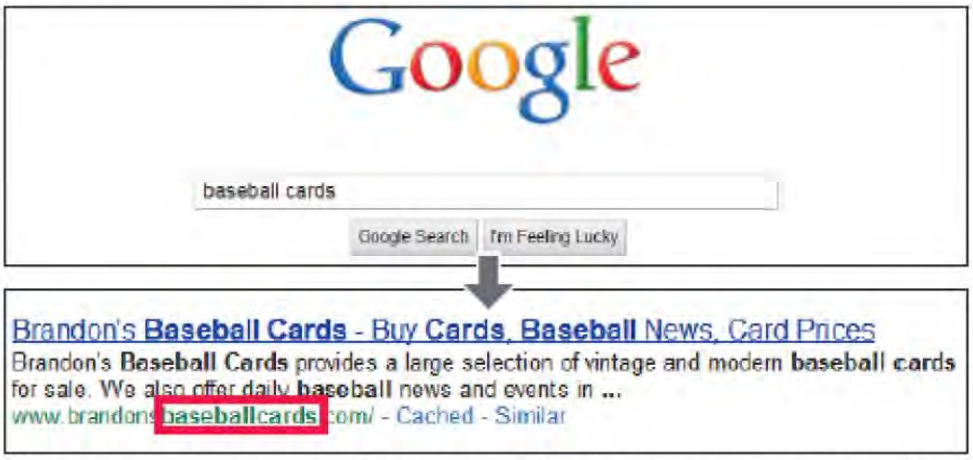
Hey, I found this article on the rarest baseball cards. These cards are FTW

The link is http://www.brandonsbaseballcards.com/articles/ten-rarest-baseball-cards.htm

(2) The highlighted words above could inform a user or search engine what the target page is about before following the link.

confusing and unfriendly. Users would have a hard time reciting the URL from memory or creating a link to it. Also, users may believe that a portion of the URL is unnecessary, especially if the URL shows many unrecognizable parameters. They might leave off a part, breaking the link.

Some users might link to your page using the URL of that page as the anchor text. If your URL contains relevant words, this provides users and search engines with more information about the page than an ID or oddly named parameter would (2).



Brandon's **Baseball Cards** - Buy **Cards**, **Baseball** News, Card Prices
Brandon's **Baseball Cards** provides a large selection of vintage and modern **baseball cards** for sale. We also offer daily **baseball** news and events in ...
www.brandons**baseballcards**.com/ - Cached - Similar

(3) A user performs the query [baseball cards]. Our homepage appears as a result, with the URL listed under the title and snippet.

Lastly, remember that the URL to a document is displayed as part of a search result in Google, below the document's title and snippet. Like the title and snippet, words in the URL on the search result appear in bold if they appear in the user's query (3). To the right is another example showing a URL on our domain for a page containing an article about the rarest baseball cards? The words in the URL might appeal to a search user more than an ID number like "www. brandonsbaseballcards.com/article/102125/" would.

4. **Make your site easier to navigate:**

The navigation of a website is important in helping visitors quickly find the content they want. It can also help search engines understand what content the webmaster thinks is important. Although Google's search results are provided at a page level, Google also likes to have a sense of what role a page plays in the bigger picture of the site.

All sites have a home or "root" page, which is usually the most frequented page on the site and the starting place of navigation for many visitors. Unless your site has only a handful of pages, you should think about how visitors will go from a general page (your root page) to a page containing more specific content. Do you have enough pages around a specific topic area that it would make sense to create a page describing these related pages (e.g. root page -> related topic listing -> specific topic)?

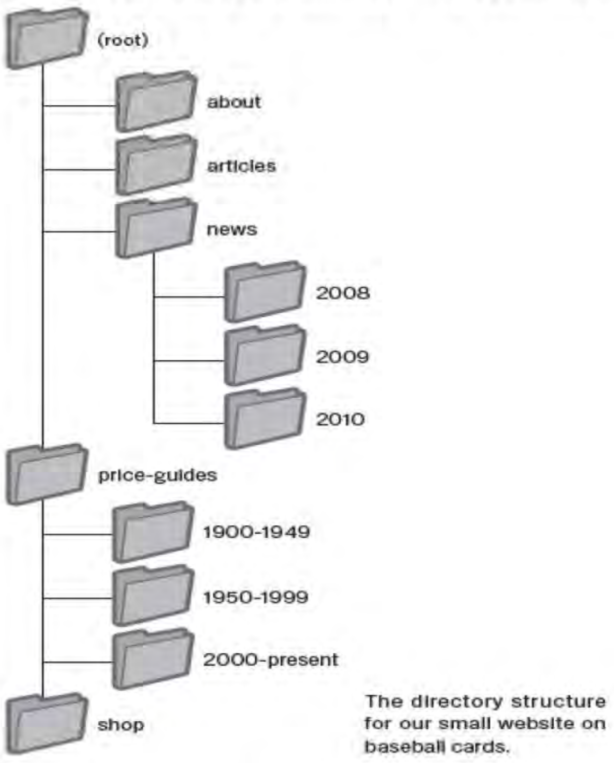


Figure 12.5 Directory Structure of small website baseball cards

A breadcrumb is a row of internal links at the top or bottom of the page that allows visitors to quickly navigate back to a previous section or the root page as shown in fig.12-6. Many breadcrumbs have the most general page (usually the root page) as the first, left-most link and list the more specific sections out to the right.

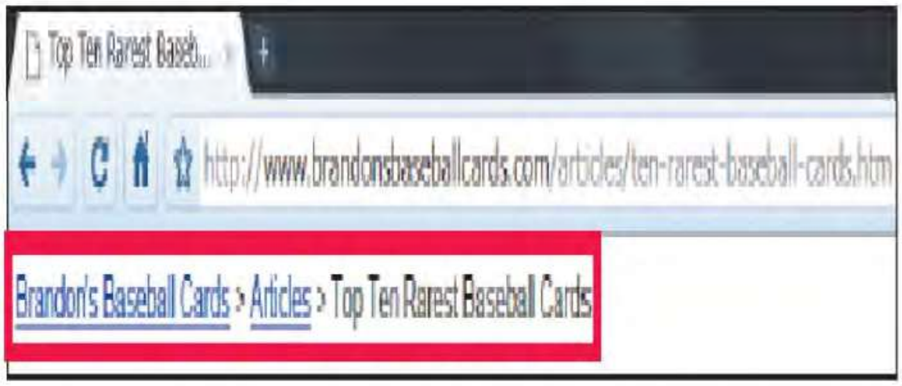
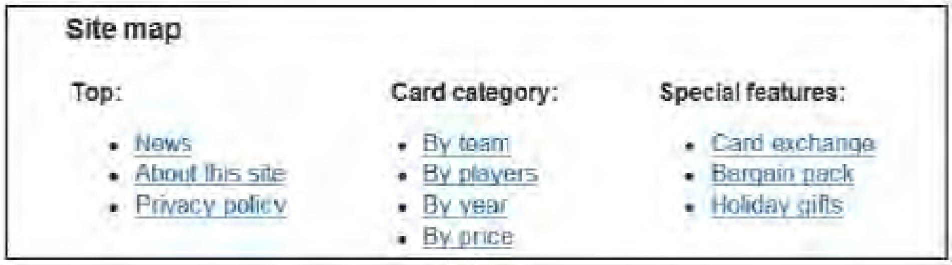

Top Ten Rarest Baseb...
http://www.brandonsbaseballcards.com/articles/ten-rarest-baseball-cards.htm
Brandon's Baseball Cards > Articles > Top Ten Rarest Baseball Cards

Figure 12.6 Breadcrumb Links appearing on a dipper article page of website


Site map

Top:
- News
- About this site
- Privacy policy

Card category:
- By team
- By players
- By year
- By price

Special features:
- Card exchange
- Bargain pack
- Holiday gifts

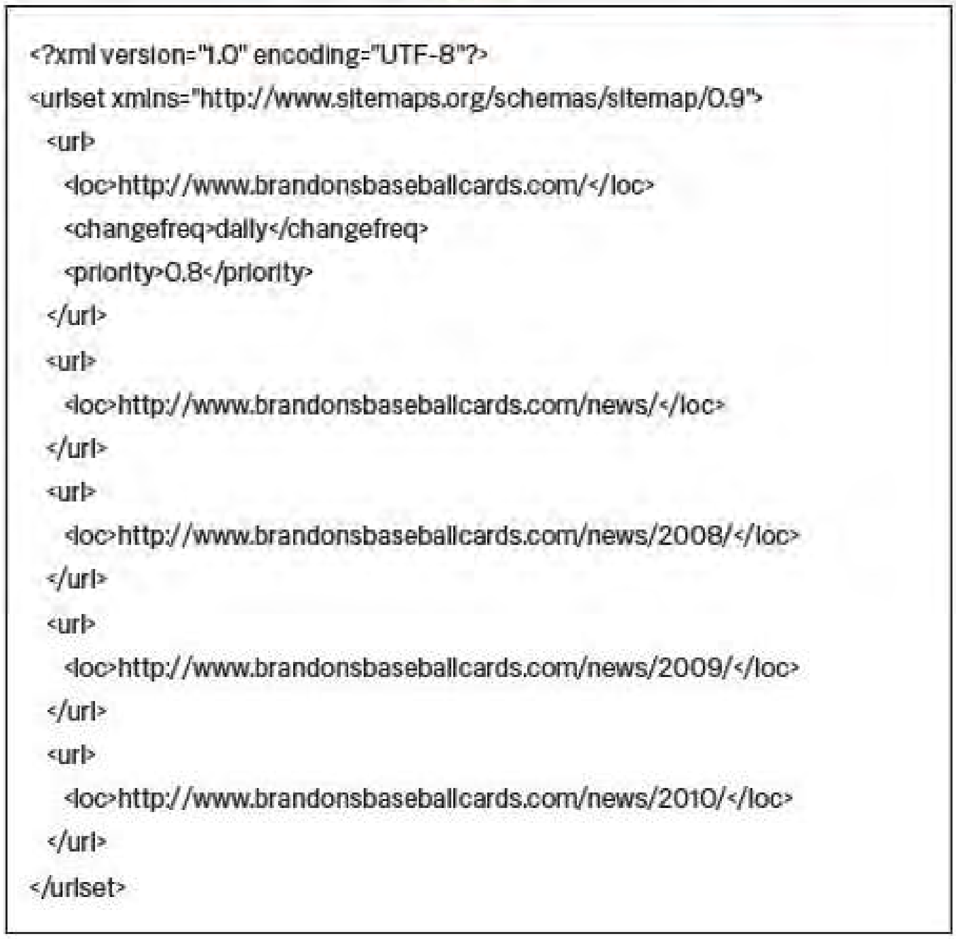

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.brandonsbaseballcards.com/</loc>
    <changefreq>daily</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://www.brandonsbaseballcards.com/news/</loc>
  </url>
  <url>
    <loc>http://www.brandonsbaseballcards.com/news/2008/</loc>
  </url>
  <url>
    <loc>http://www.brandonsbaseballcards.com/news/2009/</loc>
  </url>
  <url>
    <loc>http://www.brandonsbaseballcards.com/news/2010/</loc>
  </url>
</urlset>
```

Figure 12.7 Example of an HTML site-map and XML sitemap

5. Prepare site map

We have to prepare two site maps one for users and another for search engine. A site map (lower-case) is a simple page on your site that displays the structure of your website, and usually consists of a hierarchical listing of the pages on your site. Visitors may visit this page if they are having problems finding pages on your site. While search engines will also visit this page, getting good crawl coverage of the pages on your site, it's mainly aimed at human visitors.

An XML Sitemap (upper-case) file, which you can submit through Google's Webmaster Tools, makes it easier for Google to discover the pages on your site. Using a Sitemap file is also one way (though not guaranteed) to tell Google which version of a URL you'd prefer as the canonical one (e.g. http://brandonsbaseballcards.com/ or http:// www.brandonsbaseballcards.com/;

12.5 Effective Content Writing Plan

The effective content writing plan includes following factors:

1. **Write better anchor text**

   Anchor text is the clickable text that users will see as a result of a link, and is placed within the anchor tag <a href="..."></a>.

   This text tells users and Google something about the page you're linking to. Links on your page maybe internal—pointing to other pages on your site—or external—leading to content on other sites. In either of these cases, the better your anchor text is, the easier it is for users to navigate and for Google to understand what the page you're linking to is about.

2. **Optimize your use of images**

   Images may seem like a straightforward component of your site, but you can optimize your use of them. All images can have a distinct filename and "alt" attribute, both of which you should take advantage of. The "alt" attribute allows you to specify alternative text for the image if it cannot be displayed for some reason.

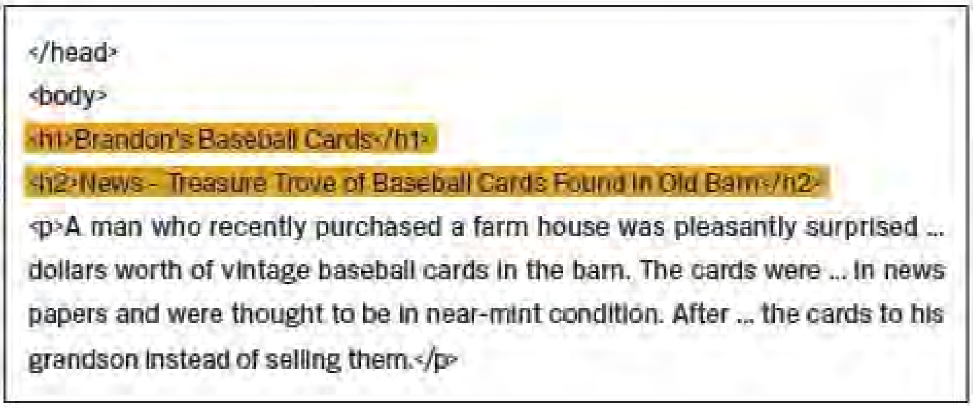
   Why use this attribute? If a user is viewing your site on a browser that doesn't support images, or is using alternative technologies, such as a screen reader, the contents of the alt attribute provide information about the picture.

   Another reason is that if you're using an image as a link, the alt text for that image will be treated similarly to the anchor text of a text link. However, we don't recommend using too many images for links in your site's navigation when text links could serve the same purpose. Lastly, optimizing your image filenames and alt text makes it easier for image search projects like Google Image Search to better understand your images.

And store files in specialized directories and manage them using common file format. Use commonly supported file types - Most browsers support JPEG, GIF, PNG, and BMP image formats.

3. **Use heading tag appropriately**

Heading tags (not to be confused with the <head> HTML tag or HTTP headers) are used to present structure on the page to users. There are six sizes of heading tags, beginning with <h1>, the most important, and ending with <h6>.



```
</head>
<body>
<h1>Brandon's Baseball Cards</h1>
<h2>News - Treasure Trove of Baseball Cards Found in Old Barn</h2>
<p>A man who recently purchased a farm house was pleasantly surprised ...
dollars worth of vintage baseball cards in the barn. The cards were ... in news
papers and were thought to be in near-mint condition. After ... the cards to his
grandson instead of selling them.</p>
```

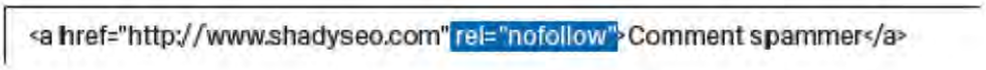
4. **Make Effective use of Robot.txt**

A "robots.txt" file tells search engines whether they can access and therefore crawl parts of your site. This file, which must be named "robots.txt", is placed in the root directory of your site.

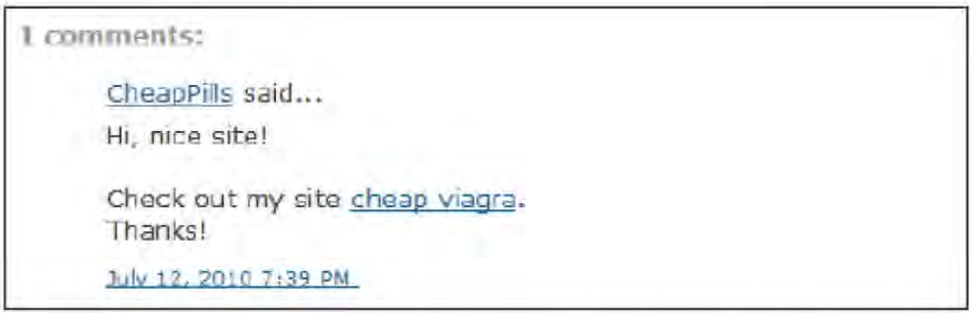
You may not want certain pages of your site crawled because they might not be useful to users if found in a search engine's search results. If you do want to prevent search engines from crawling your pages, Google Webmaster Tools has a friendly robots.txt generator to help you create this file. Note that if your site uses subdomains and you wish to have certain pages not crawled on a particular subdomain, you'll have to create a separate robots.txt file for that subdomain. For more information on robots.txt, we suggest this Webmaster Help Center guide on using robots.txt files.

5. **Be aware of rel="nofollow" for links**

Setting the value of the "rel" attribute of a link to "nofollow" will tell Google that certain links on your site shouldn't be followed or pass your page's reputation to the pages linked to. No following a link is adding rel="nofollow" inside of the link's anchor tag.



```
<a href="http://www.shadyseo.com" rel="nofollow">Comment spammer</a>
```



1 comments:

CheapPills said...

Hi, nice site!

Check out my site cheap viagra.
Thanks!

July 12, 2010 7:39 PM

When would this be useful? If your site has a blog with public commenting turned on, links within those comments could pass your reputation to pages that you may not be comfortable vouching for. Blog comment areas on pages are highly susceptible to comment spam.

12.6 Achieving High Ranking

High ranking means you have to put your query on search engine. Your result should be displayed at web page in top ranking i.e. result link contains higher priority as compare to other link. To achieve high ranking we should follow below tips:

1. Avoid using a lot of images or Flash content on your website, because this kind of content is not visible to search engines, and won't help your ranking. It goes the same for JavaScript.
2. Images and flash objects are fun and eye appealing, but always use them with texts that describe them.
3. Make sure your website is fast. A lot of images, Flash and Java objects will slow down the loading time, another issue taken in consideration by Google.

4. Avoid having duplicate pages (the same content on more than one page), because Google will only index them once.
5. Use a keyword tool (like Google AdWords Keyword tool) to find out what are the most valuable keywords for the entire website and for each page individually: those that have a high number of searches and low competition.
6. The URL matters, so have a domain name that's related to your keywords and content. You can buy a domain or get one for free from dot.tk, and use it for free with Snack Websites.
7. Your keywords should appear in your website's description. Go to *manage website -> Website description*, and write a short description of your website and include your most valuable keywords.
8. Write a page name and set a Meta title (to set the *Meta title*, click *more options*- bottom of page) for each one of your pages. Write descriptive page names and use different Meta titles for pages, and make sure they are relevant for the content you have on the page. Google understands that each page is different, and will help each individual page "rank" in search results.
9. It is important to write posts that are 250 words or more for all pages you wish to see "rank" in Google. For each 250 words of text, try to include 2-3 mentions of the keyword you are targeting for that page.
10. Google loves frequently updated websites and fresh content, so update your blog once/twice a week. You should add a blog section to your website, if you haven't already.
11. Have a lot of content on your website. Write a lot of posts and readers will come.
12. Write your posts yourself, and don't just copy from different websites or blogs. Search engines appreciate unique content.

13. Try to have other websites link to your blog/ website, because the number of links directing to your website influences your rank in searches.

14. Its great if the anchor text linking to your website or a certain page from your website isn't something like "click here", but a keyword. If you link within your website, from one page to another, follow the same principle.

15. Share your website on social media: Facebook, Twitter, Google+.

12.7 SEO Analysis Intervals

The user rarely knows the exact web address of a company. Furthermore, most users search just for a specific product, or just a kind of product. After the user types the desired keywords into the search engine's input field, the search engine responds with a long list of Websites connected with the input keywords. Naturally, people search for information only among the top 5 to 10 positioned websites.

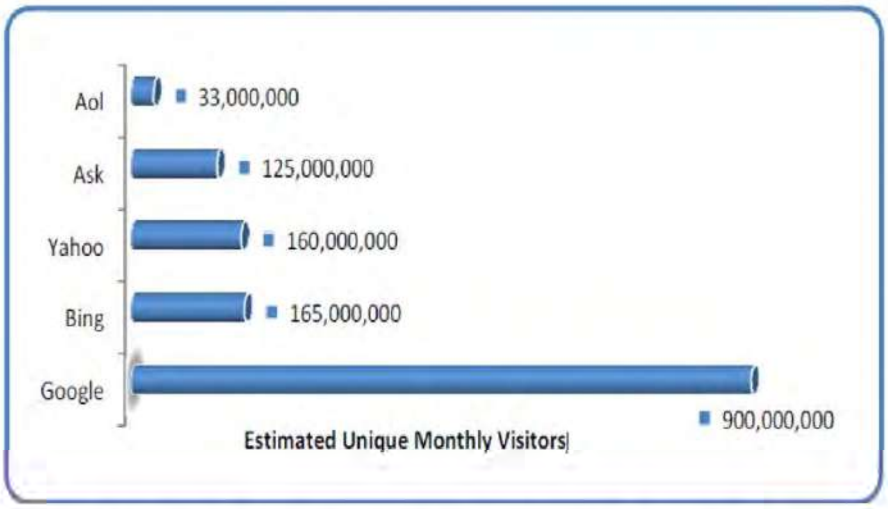


Figure 12.8 Estimated unique monthly visitors of most popular search engines

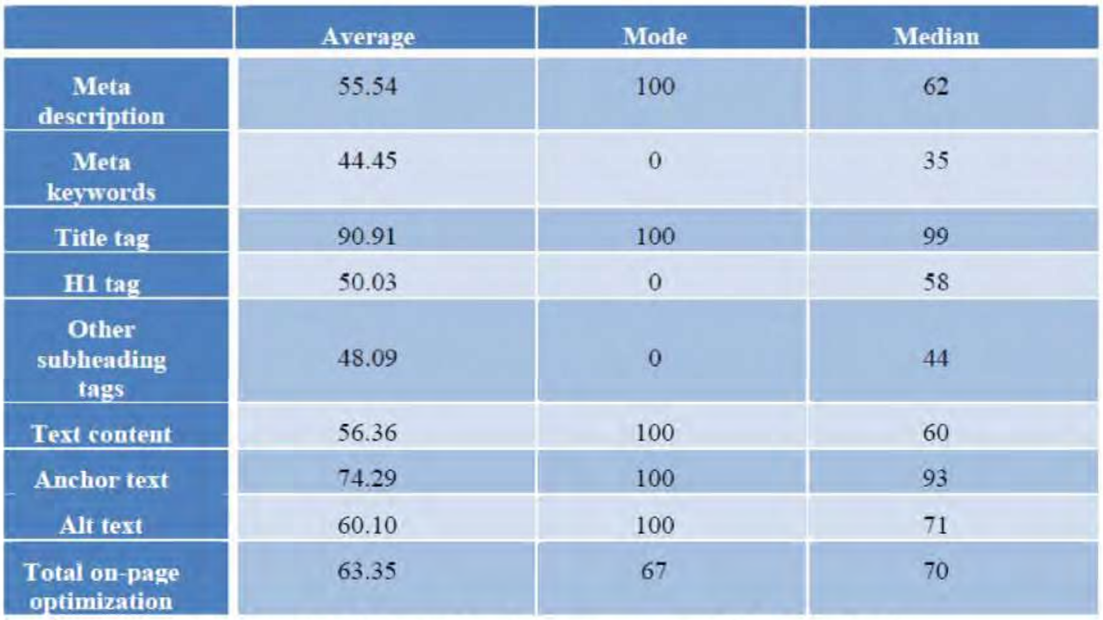
We used three online SEO analysis tools - lipperhey, alexea, and seomoz and checked 495 Websites from 7 different countries: Serbia, Croatia, Bosnia and Herzegovina, Australia, Canada, USA, and UK. We chose the top five Websites from the following 23 categories: toy stores, book stores, furniture, cars, clothes, confectionery, cosmetics, sporting goods departments, printing offices, newspapers, foreign language schools, watch stores, eyeglasses stores, home care, dental clinics, bicycle stores, health food stores, tourist agencies, banks, hotels, flower shops, computer shops. The aim is to realize trends of on-page optimization among the top ranked Websites in practice.

The obtained data can serve as a recommendation on what on-page SEO steps SEOs (people who work on SEO) should pay more attention to. These recommendations could also be very useful for increasing usability, since as already mentioned, many on-page optimization factors also influence usability. In addition to a future off-page analysis and their comparison, this data can give us information on the role of on-page optimization within the complete SEO process.

Let us first explain the sample of Websites used in our investigation. First, we made a list of 23 random categories among the most frequently used search terms. Then, for four to five of the above-mentioned countries, we analyzed the top five Websites for each category using one of the three SEO analysis tools listed above. Among many other available data, we were interested in the percentage of optimization of the following HTML elements:

- Meta description,
- Meta keywords,
- Title tag,
- H1 – tag,
- Other subheading tags,
- Text content,
- Anchor text,
- Alt text, and
- The total optimization of the Website.

Besides, we were interested in the fact if the analyzed Websites had an XML site map. The obtained results are given in fig. 12-9. The first column of the table represents the average values of every above-listed on-page SEO issue for all Websites from the survey, i.e. the sum of all obtained optimization values for each issue divided by 495. The second column in the table represents the mode values and other optimization issues. The mode in a list of numbers refers to the list of numbers that occur most frequently. The so-called 'middle values' or medians of the listed optimization issues are given in the third column. In order to obtain the median of a list of values, one has to sort the list in an increasing order. Then, if the totals of the list are odd, the median is just the middle entry. Otherwise, if the totals of the list are even, there appear to be two middle entries. To get the median one, has just to sum them and divide by two.



	Average	Mode	Median
Meta description	55.54	100	62
Meta keywords	44.45	0	35
Title tag	90.91	100	99
H1 tag	50.03	0	58
Other subheading tags	48.09	0	44
Text content	56.36	100	60
Anchor text	74.29	100	93
Alt text	60.10	100	71
Total on-page optimization	63.35	67	70

Figure 12.9 Result of the on-page SEO analysis

Figure 12.10 shows the distribution of the percentage of title tag optimization. As we can see, the majority of points are in the interval of 80-100%.

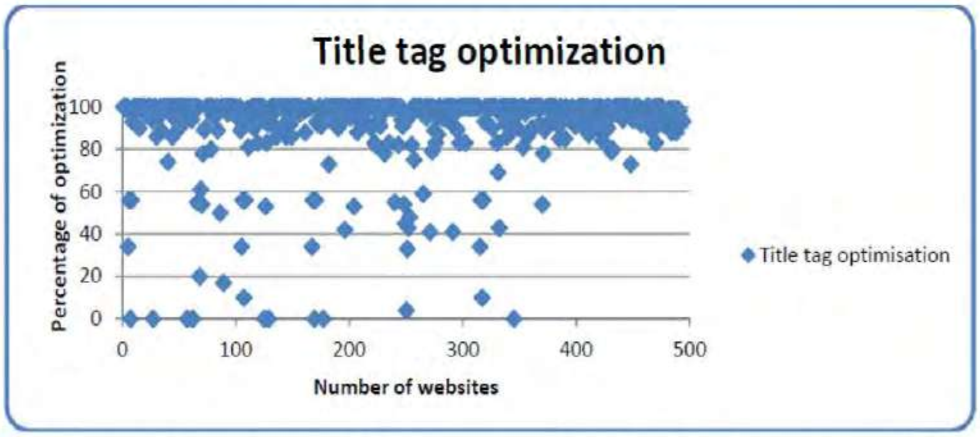


Figure: 12.10 the distribution of title tag optimization

Anchor text is also well optimized in a high average percentage, 74.29%. Figure 12-11 shows the distribution of the anchor text optimization. This time, the sample is more dissipative than in the case of title tag optimization.

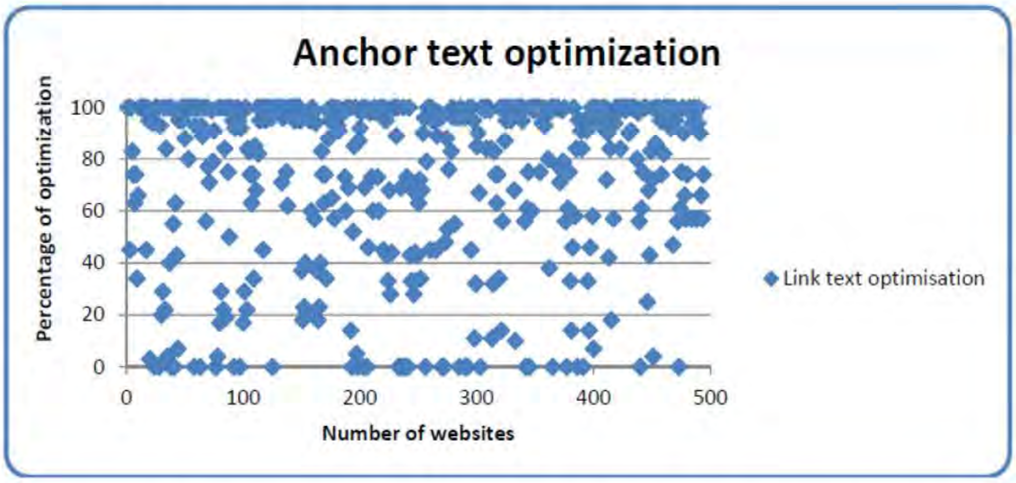


Figure 12.11the distribution of anchor text optimization

The Meta keywords tag holds the lowest percentage of optimization, 44.45%. But we will not pay too much attention to this data, since, nowadays, the majority of search engines ignore the Meta keywords tag.

12.8 Self Learning Exercise

Q.1 SEO stands for __________.

(A) Search Engine Operation (B) Search Entry Optimization

(C) Search Engine Optimization (D) None of these

Q.2 Search Engine Optimization is the process of ______ a website in a search engine's search results.

(A) Generating Cache files (B) Affecting the visibility

(C) Getting Meta tag (D) none of these

Q.3 Writing repeating keywords in the Meta tags and unrelated keyword is called as __________.

(A) Meta tag stuffing (B) Meta tag hijacking

(C) Meta tag cloaking (D) none of these

Q.4 Which is the process of fetching all the web pages linked to a web site?

(A) Crawling	(B) Processing
(C) Calculating Relevancy	(D) Indexing

12.9 Summary

A web search engine is a tool that allows user to find the information on the World Wide Web. We discussed the block architecture of a search engine. It is divided into two parts: first is indexed processing and second is query processing. Keywords are what help your site get recognized by Internet search engines, and as a result, help would-be customers find your Internet presence. Search engine optimization is a methodology of strategies, techniques and tactics used to increase the amount of visitors to a website by obtaining a high-ranking placement in the search results page of a search engine. SEO are classified into two types: on-page optimization and off-page optimization. Various steps were discussed to design a web page using SEO. Also include the effective writing plan and Different SEO analysis intervals.

12.10 Glossary

SEO: Search Engine Optimization.

HTTP: Hyper Text Transfer Protocol.

JPEG: Joint Picture Expert Group.

12.11 Answers to Self-Learning Exercise

Ans.1: C	**Ans.2:** B
Ans.3: A	**Ans.4:** A

12.12 Exercise

Q.1 What do you mean by web crawling? Explain.

Q.2 How we can achieve high ranking in SEO? Explain in detail.

Q.3 Explain the architecture of search engine.

Q.4 What is search engine? How does it works?

Q.5 How we can improve the structure of URL? Explain.

Q.6 What are the different types of activities performed by search engine?

Q.7 Differentiate between black hat SEO and White hat SEO.

Q.8 Explain the various steps involved in effective content writing plan.

12.13 Answers to Exercise

Ans.1, Ans.2, Ans.3 and **Ans.4** are discussed in chapter in details. See respective contents.

References and Suggested Readings

1. Papazoglou, Web Services: Principles and Technology (2nd edition); ISBN: 978-0-273-73216-7, Prentice Hall, 2012.
2. Alonso, Casati, Kuno and Machiraju, Web Services: Concepts, Architectures and Applications; ISBN: 3540440089, Springer, 2004.
3. Cerami, Web Services Essentials; ISBN: 0596002246, O'Reilly, 2002.
4. Seomoz Analytics Tool. (2012). Retrieved December 25, 2012 from http://www.seomoz.org.

UNIT 13
Web Content Management System

Structure of the Unit

13.0 Objective

The objective of this unit is to make the students aware with how the web contents are developed and managed online and what is the software which provides such kind of facility to organize contents over web? In this unit, student will know about some important open source software for this purpose which is called web content management system.

13.1 Introduction

Today, information is getting important for all. Over internet, a huge amount of information is available in different websites. In website, this information is available in contents. Different website may have different propose. According to the purpose of website, web contents are also accordingly kept on it. To manage the contents, are important task because these are presented to the user/s where they could easily get it through website. Content management focuses to the system and processes for creating, managing, publishing and archiving the information. Complete processing steps are managed by a system that is called content management system (CMS) which provides the necessary infrastructure to effectively contribute the contents and collaborate throughout these lifecycle of process.

Contents in a website are the basic and necessary component for preparation and publication of a website which may be very simple or complex. Contents of a website may need to manage as per requirement. For CMS, three major stages are focused: content, managed and publication stage. In general, a CMS is a web application that runs on web server to help facilitate creating a website.

A Web Content Management System (WCMS) is a web application which provides the facility to a group of users, usually from different divisions in an organization, to collaboratively maintain and organize the content of a website in an effective manner. Today web content management systems are very much popular and have grown in importance as more and more organizations are communicating and publishing their information via the web. Some open source software which helps to manage the contents is described below:

13.2 WordPress

13.2.0Introduction

WordPress is one of the most popular open source web content management systems and website creation tool, available online. It is freely available, we can download it and according to our needs, we can modify, store, manage and update the contents and lastly publish and host it. It is written in PHP. In general, it is popular to develop blogging sites.

For running a WordPress locally, we require PHP, MySql and Apache Web server on local system but it can also installed on a web server that is either part of an Internet hosting service or a network host.

There are many features of WordPress. Some of them are given below:

13.2.1Features of WordPress

WordPress has several features. Some of them are described below:-

Simplicity

WordPress is simple to use. The simplicity makes it possible for us to understand the process, work on it, to get online and publish it quickly. Everything is managed systematically in the way of getting the website and contents are well organized.

Flexibility

WordPress is very much flexible. Using this CMS, we can create any type of website. We can use themes and extend the functionality by plugins to give look and feel to our website attractive. We can even build our own application.

Easy Publishing

Creating content with WordPress and publishing on real time is very easy. We can create Posts and Pages, format them easily, inserting media, and with the click of a button, the contents are easily live and on the web.

Publishing Tools

WordPress helps to makes it easy for users to manage the content/s. Create drafts, schedule publication, and look at the post revisions. We can make the content/s public or private, and secure the posts and pages with a password.

User Management

In WordPress, everyone requires the same access to the website. Website administrator manages the site, users, contents, contributors and subscribers. There are different types of contributors in the website and they are the part of the website community.

Media Management

It is the tool for managing variety of media files and folder, through which we can easily upload, organize, drag and drop and also manage different media files (audio, video, text etc.) on our website. In addition to it, we can also manage the contents by adding text, captions and inserting titles and images and making galleries.

W3C Compliance

Every piece generated in WordPress is in completely with W3C compliance standard. This means that WordPress website is compatible with today's browser and compatible with the next generation browser.

Simple Theme System

WordPress gives two themes by default, but there is a theme directory with thousands of themes for you to build a beautiful website. We can also develop our theme and upload it to our website and also customize it. It only takes a few seconds to give your website a complete change.

Rich in Plugins

WordPress is very much rich in plugins having full of features for every user. There is also a plugin directory where thousands of plugins are available. According to our needs, we download and use it.

Built-in Comments

A comment in WordPress is an important feature which provides a space for your friends and followers to engage with your content in the blog developed by yourself. The comment tool gives you everything you need to be a forum for discussion and to moderate that discussion.

Search Engine Optimized

WordPress is optimized for search engines with plenty of SEO plugins available to makes SEO website simple and increase the popularity of the website on search engine hit.

Multilingual

WordPress is multilingual web content system which is available in more than 70 languages. If you or the person is planning to build the website, you can prefer to use WordPress in a language other than English, that's easy to do.

Easy Installation and Upgradation

Installation and upgradation is very easy in WordPress. We can upload and install it by using an FTP program. Another option to upload and install it by web hosts which offer one-click WordPress installers.

Importers

WordPress comes with importers for blogger, Live Journal, Movable Type, TypePad, Tumblr.

Personal Data

In WordPress, no one have the access to our content.

Freedom

WordPress is licensed under the GPL, created to protect your software freedoms. You are free to use it in any way, install and modify it, and also distribute it. Software freedom is the foundation that it is built on.

Community

As the most popular open source CMS on the web, WordPress has a vibrant and supportive community. We can ask a question on the support forums and get help from experts and learn more about WordPress, read blogs posts and tutorials about WordPress.

Contribution

As we know that WordPress is a free community to share the ideas and improve the WordPress website. Everyone who wants to know about the query they can ask for and others who know about the query can share. Even they can also develop some new ideas to improve the WordPress website which may be useful for user community.

13.2.2 Advantages

Most of the website today is being developed in WordPress because of its simplicity, flexibility and helping community because of several advantages. Some of them are given below:

- It is an open source platform and freely available for use.
- Customization is easy according to the user's needs.
- Easy user management.
- Media files can be uploaded easily and quickly.
- There are many plugins and templates available for free. Users can customize the various plugins as per their need.
- Easy design using CSS files as per the users need.
- Simple WYSIWYG editor (What You See Is What You Get) allows users to directly work on the layout of document without having a layout command.
- It offers several SEO tools which makes on-site SEO simple.

13.2.3 Disadvantages

In addition to several advantages, WordPress also has some disadvantages:

- Use of many plugins can make the website heavy to load and run.
- PHP knowledge is required to make modifications or changes in the WordPress website.

- Sometimes software needs to be updated to keep the WordPress up-to-date with the current browsers and mobile devices. Updating WordPress version leads to loss of data, so it a backup copy of the website is required.
- Modifying and formatting the graphic images and tables is difficult.

13.2.4 Requirements for WordPress

To run WordPress we must have the following supporting tools:

- PHP version 7 or greater
- MySQL version 5.6 or greater OR MariaDB version 10.0 or greater
- HTTPS support

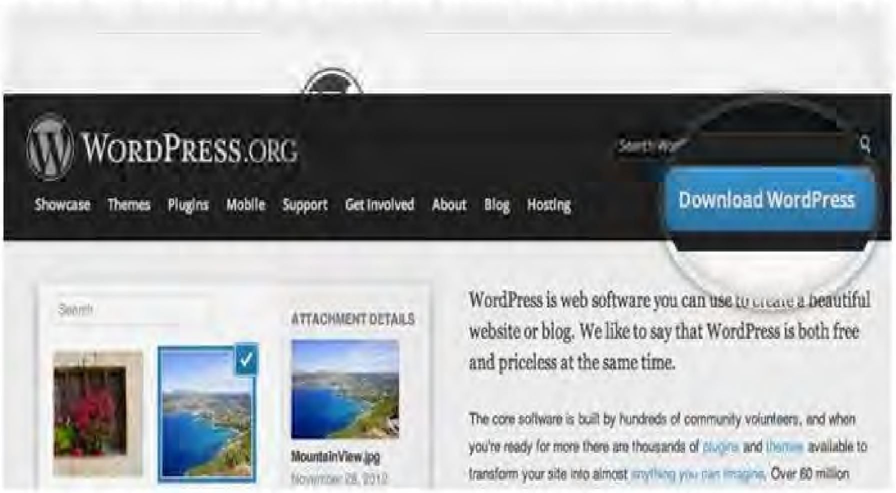
13.2.5 Installation of WordPress

Following steps are used to install WordPress manually:

Step 1:	Download the WordPress installation package
Step 2:	Upload the WordPress files to your server
Step 3:	Creating MySQL Database
Step 4:	Installation process

Download the WordPress installation package

To start the installation process, first we need to download WordPress from its official download page. For the security concern etc., we must download and install latest stable version of WordPress from its website (www.WordPress.org).



WordPress is web software you can use to create a beautiful website or blog. We like to say that WordPress is both free and priceless at the same time.

The core software is built by hundreds of community volunteers, and when you're ready for more there are thousands of plugins and themes available to transform your site into almost anything you can imagine. Over 60 million

Figure-13.0 Download WordPress Page

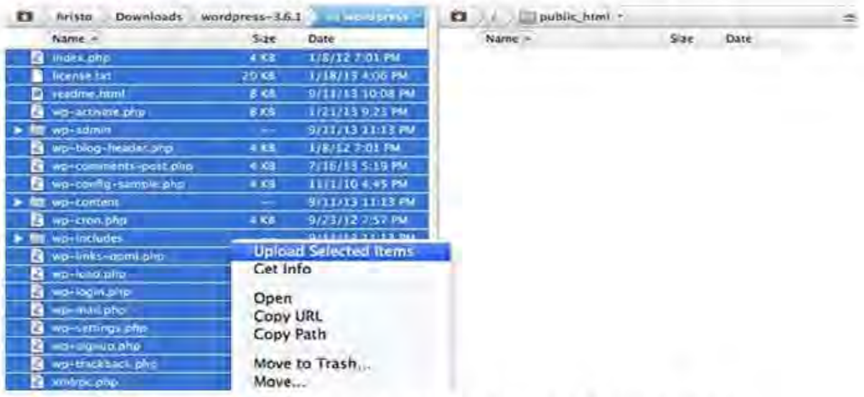
As you click on the Download button for the latest WordPress version, the installation package will be saved to your hard disk. Here we have to locate the path where the installation package will be stored and extract it to a new folder.

Upload the WordPress Files to Your Server

Now, you need to upload the extracted files and folders to your web server. The one simple way to upload the installation files is through FTP.

Note: If you are installing the WordPress in your main domain (i.e. www.mydomain.com), you need to upload the extracted files into the public_html folder.

Note: Another easy method is that, you can also download WAMP or XAMPP for windows and LAMP package for Linux Operating system. After downloading one of the package as according to the operating system you have, install it. The installation will be done by default into a C:\ drive and WAMP or XAMPP. We have to put all the extracted WordPress files into www directory of the WAMP or XAMPP. WAMP or others are the combination of three main software packages - PHP, MySql and Apache. Start the WAMP and follow the steps.



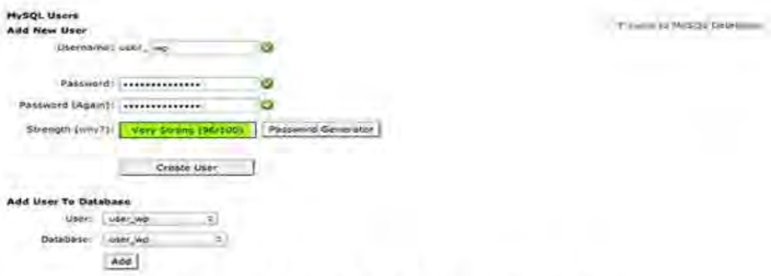
Name	Size	Date
index.php	4 KB	1/8/12 7:01 PM
license.txt	20 KB	1/18/13 4:06 PM
readme.html	8 KB	9/11/13 10:08 PM
wp-activate.php	8 KB	1/21/13 9:23 PM
wp-admin	--	9/11/13 11:13 PM
wp-blog-header.php	4 KB	1/8/12 7:01 PM
wp-comments-post.php	4 KB	7/16/13 5:19 PM
wp-config-sample.php	4 KB	11/1/10 4:45 PM
wp-content	--	9/11/13 11:13 PM
wp-cron.php	4 KB	9/23/12 2:57 PM
wp-includes		
wp-links-opml.php		
wp-load.php		
wp-login.php		
wp-mail.php		
wp-settings.php		
wp-signup.php		
wp-trackback.php		
xmlrpc.php		

Name	Size	Date

Figure-13.1 Uploading WordPress files

Create a MySQL Database for WordPress

Now, you need to create a MySQL database with full permissions. Once you create MySQL Database and User, it also requires the database name, database username and password as you just created.



MySQL Users

Add New User

Username: user_wp

Password: ••••••••••••••

Password (Again): ••••••••••••••

Strength (why?): Very Strong (96/100) | Password Generator

Create User

Add User To Database

User: user_wp

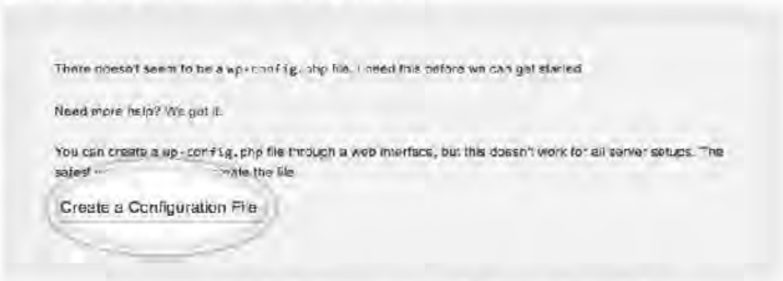
Database: user_wp

Add

Figure-13.2 Database creation

Installation process

Now this is the time to start the installation process. Here you have to little bit care about the message, regarding wp-config.php configuration file. Just click on the Create a Configuration File button to proceed.



There doesn't seem to be a `wp-config.php` file. I need this before we can get started.

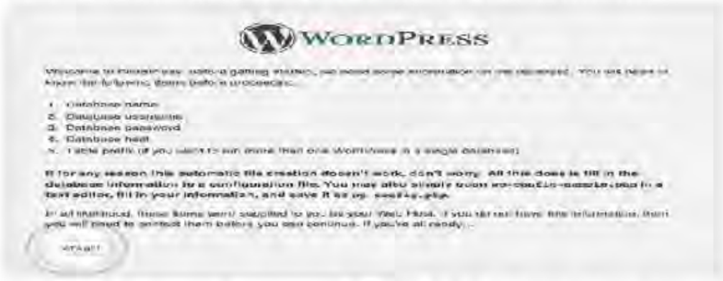
Need more help? We got it.

You can create a `wp-config.php` file through a web interface, but this doesn't work for all server setups. The safest [illegible] ate the file.

Create a Configuration File

Figure-13.3 Create configuration file

On this page, you will see a message, asking you to prepare the necessary information for the installation. Simply press the Go! Button.



WordPress

[illegible]

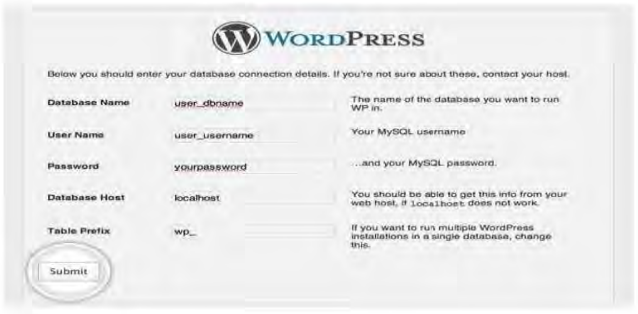
1. Database name
2. Database username
3. Database password
4. Database host
5. Table prefix (if you want to run more than one WordPress in a single database)

[illegible]

[illegible]

Figure-13.4 requirement fulfil in installation portion

In the next WordPress will ask for to create MySQL database and press the Submit button

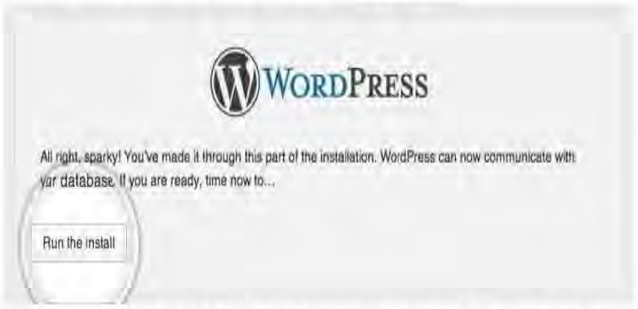

WORDPRESS

Below you should enter your database connection details. If you're not sure about these, contact your host.

Database Name	user_dbname	The name of the database you want to run WP in.
User Name	user_username	Your MySQL username
Password	yourpassword	...and your MySQL password.
Database Host	localhost	You should be able to get this info from your web host, if localhost does not work.
Table Prefix	wp_	If you want to run multiple WordPress installations in a single database, change this.

Figure-13.5 Database name creation

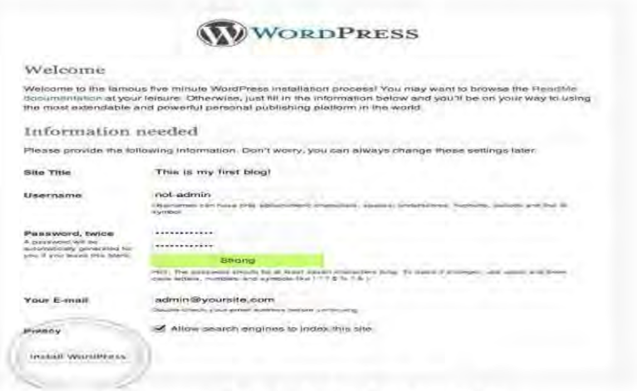
After click on submit button, WordPress checks the necessary settings. If we have entered necessary information correctly then we will get a confirmation screen and click on the Run the Install button to further process.


WORDPRESS

All right, sparky! You've made it through this part of the installation. WordPress can now communicate with your database. If you are ready, time now to...

Figure-13.6 Confirm and run the install

The next screen is all about for administrative username and password and title of a new website.


WORDPRESS

Welcome

Welcome to the famous five minute WordPress installation process! You may want to browse the ReadMe documentation at your leisure. Otherwise, just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

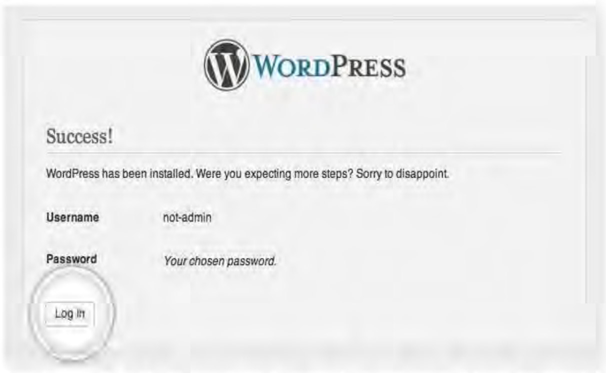
Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title	This is my first blog!
Username	not-admin [illegible]
Password, twice [illegible]	············ ············ Strong [illegible]
Your E-mail	admin@yoursite.com [illegible]
Privacy	✓ Allow search engines to index this site.

Figure-13.7 Admin details

THE INSTALLATION PROCESS IS COMPLETED. NOW YOU CAN LOGIN BY USERNAME AND PASSWORD AND ACCESS THE WEBSITE.



Success!

WordPress has been installed. Were you expecting more steps? Sorry to disappoint.

Username not-admin

Password *Your chosen password.*

Figure-13.8 Success and access the website

13.2.6 Working on WordPress

Before starting in newly created Wordpress website, you should first log into as an administration. As you do it, you will see the Dashboard of the website. Dashboard displays the overview of the website and has a collection of gadgets related to the blog or website. We can customize this by our needs.

The snapshot of Dashboard is shown below for your understanding. It contains several sub-fields that are described below:

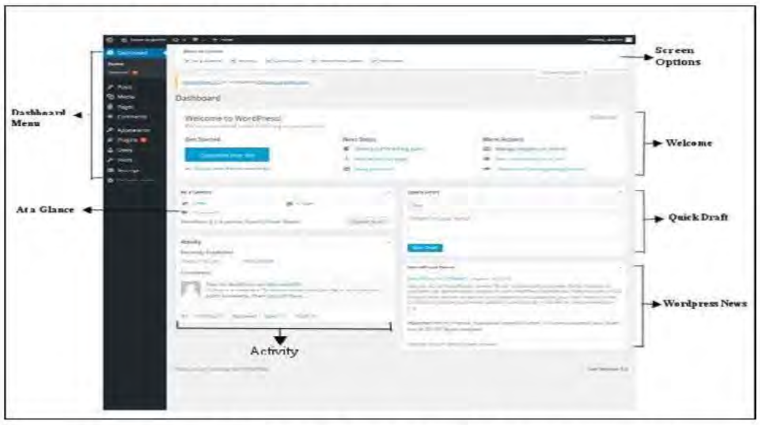


Figure-13.9 Dashboard of blog / website

Dashboard Menu

Dashboard provides the navigational menu which is related to the posts, media library, pages, comments, appearance options, plugins, users, tools and settings on the left side.

Screen Options

The dashboard contains different types of widgets which can be shown or hidden on some screens. It contains check boxes to show or hide screen options and also allows us to customize sections on the admin screen.

Welcome

This provides the Customize Your Site button, allowing the customization of current WordPress theme.

Quick Draft

It is a mini post editor which allows writing, saving and publishing a post from admin dashboard. It includes the title for the draft, some notes about the draft and save it as a Draft.

News

News widget shows the latest news such as latest software version, updates, alerts, news regarding the software etc.

Activity

This widget includes latest comments on your blog, recent posts and recently published posts. It allows you to approve, disapprove, reply, edit, or delete a comment. It also allows you to move a comment to spam.

At a Glance

This section gives an overview of your blog's posts, number of published posts and pages, and number of comments. When we click on the links, we move to the respective screen. It also displays the current version of running WordPress along with the currently running theme on the site.

General Settings

WordPress general setting is used to set the basic configuration settings for the website. Following are the steps to access the general settings –

Step 1 – Click on Settings →General option in WordPress

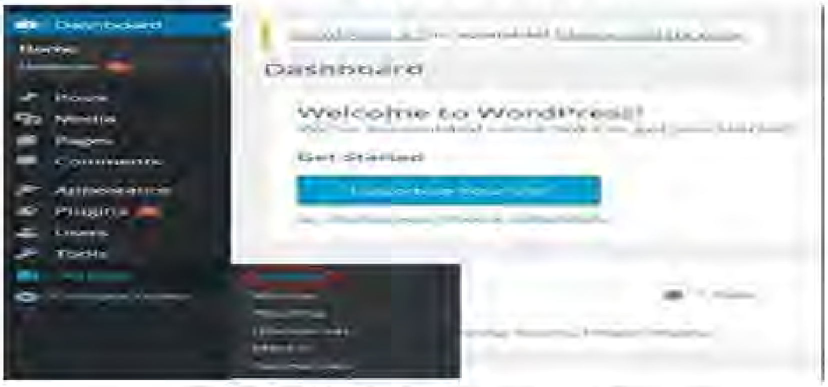


Figure-13.10 General Settings

Step 2 – The General Setting page is displayed as shown in the following snapshot.

Figure-13.11 General Setting Page fields

Fields on general settings page are briefly describes:-

- Site Title – It displays the name of the site in the template header.
- Tagline – It displays a short sentence about your site.
- WordPress Address (URL) – It is the URL of WordPress directory where your all core application files are present.
- Site Address (URL) – Enter the site URL which you want your site to display on the browser.
- E-mail Address – Enter your e-mail address which helps to recover your password or any update.
- Membership – anyone can register an account on your site after you check this checkbox.
- New User Default Role – The default role is set for the newly registered user or members.
- Timezone – Sets the time zone based on the particular city.
- Date Format – Sets the date format as you need to display on the site.
- Time Format – Sets the time format as you need to display on the site.
- Week Starts On – Select the week day which you prefer to start for WordPress calendar. By default it is set as Monday.
- Site Language – Sets the language for the WordPress dashboard.

Step-3: After entering all required details, click on Save Changes button. It saves all your general setting information.

13.2.7 Basic Components of WordPress

WordPress provides a user friendly way of creating websites. It starts with as a blogging platform. There are some basic components which help to create a website or blog efficiently. These are described as below:

13.2.7.1 Framework

A framework refers a backbone of a Theme which provides a flexible foundation to create the website/blog with speed up design. It is very similar to the interior wall of a building. It refers to design layout of blog or websites. Themes that use the same Framework often are set up very similarly. This means that information is stored in the same location and pages are set up in very similar ways.

13.2.7.2 Theme

Theme in WordPress is a set of files that provides the look and feel of your WordPress website. It creates the general layout of the site pages where you can use different widgets (plugins) as your needs. This means that it contains page contents such as graphics, headers, logos and footers. You can download the theme as your requirement from WordPress official website.

13.2.7.3 Plugin

A plug-in boosts up of your website. It is a small program, or combination of programs allowing the website to add more features and do more than it could when you first installed it. There are all kinds of free and paid plugins.

13.2.8 Self Exercise

1. Define Open Source software.
2. Write advantages and disadvantes of WordPress.
3. Write the steps to install the Wordpress.
4. What are the basic requirements for WordPress?

13.3 Drupal

13.3.0 Introduction

Drupal is also a free and powerful open source Content Management System (CMS) that allows user to organize, manage and publish our contents. It is also based on PHP environments. It comes under GNU standard i.e. General Public License, that means everyone have a freedom of downloading, sharing and modifying it for others.

13.3.1Features of Drupal

Drupal has the following features:

- Creating and managing the site is easy in Drupal.
- It provides built-in user interfaces.
- It connects your website to other sites and services using feeds, search engine connection capabilities etc.
- It is open source software hence requires no licensing costs.
- It is highly flexible, creative website to the users and display more effectively to increase the visitors.
- Drupal can publish your content on social media such as Twitter, Facebook and other social mediums.
- Drupal provides more number of customizable themes, including several base themes which are used to design your own themes for developing web applications.
- It manages content on informational sites, social media sites, member sites, intranets and web applications.

13.3.2Advantages

Drupal has several advantages. Some of them are given below:

- Drupal is a flexible Content Management System that allows to handle the different content/s like- video, text, blog, menu handling, real-time statistics etc. efficiently.

- It provides a number of templates for developing web applications to give better look and feel. So there is no need to start from scratch, if you are building simple or complicated web applications.
- It is easy to manage or create blog or website. It helps to organize structure, find and reuse content.
- It gives a number of themes which gives your website an attractive look.
- It provides more than 7000 plug-ins to boost your website. Since Drupal is an open source, you can create your own plug-ins.

13.3.3 Disadvantages

In addition of advantages of Drupal, it also provides some disadvantages.

- Drupal requires some advanced knowledge and few basic things to know about the platform to install and modify. So, it is not much easy to understand the user interface.
- Drupal is powerful but new content management system. Sometimes, it is not compatible with other software.
- Sometimes performance of Drupal degrades due to the use of several plugin etc. The website which is built using Drupal will generate big server loads and never opens with a slow internet connection.

13.3.4 System Requirements for Drupal

For the sufficient and smooth running of the Drupal website, it requires some Software as well as hardware specifications. The specifications are given below:

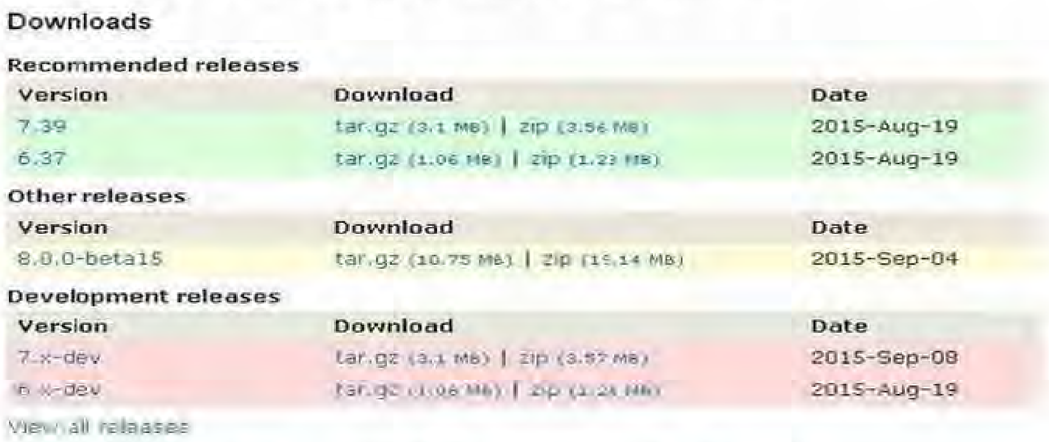
- Database : MySQL 5.1 +
- Web Server :
  - ✓ WAMP (Windows)
  - ✓ LAMP (Linux)
  - ✓ XAMP (Multi-platform)
  - ✓ MAMP (Macintosh)
  - ✓ Nginx
  - ✓ Microsoft IIS

- Operating System : Cross-platform
- Browser Support : IE (Internet Explorer 8+), Firefox, Google Chrome, Safari, Opera
- SSL (Secure Socket Layer) : A valid security certificate is required for HTTPS
- PHP Compatibility : PHP 5.2+

13.3.5 Installation of Drupal

Step-1: Download the Drupal from its official website- https://www.drupal.org/project/drupal and copy it to your web server.

Step-2: Here we have to select the latest version (zip format) in download section as shown in the below screen. The Recommended releases are the latest stable releases of either version.



Downloads

Recommended releases

Version	Download	Date
7.39	tar.gz (3.1 MB) | zip (3.56 MB)	2015-Aug-19
6.37	tar.gz (1.06 MB) | zip (1.23 MB)	2015-Aug-19

Other releases

Version	Download	Date
8.0.0-beta15	tar.gz (10.75 MB) | zip (15.14 MB)	2015-Sep-04

Development releases

Version	Download	Date
7.x-dev	tar.gz (3.1 MB) | zip (3.57 MB)	2015-Sep-08
6.x-dev	tar.gz (1.06 MB) | zip (1.24 MB)	2015-Aug-19

View all releases

Figure-13.12 Drupal Download page

Setup Wizard

It's very easy to setup Drupal on your system. The following steps describe how to setup Drupal locally on your system.

Step-1: Download the zip version file and extract it to your local computer. Rename the folder by any name that you want to use for the website.

Step-2: Drupal requires MySQL database. So create a new empty database with user/password to use. (For user as "root" and password as "root" or else you can set as per your choice).

Step-3: Open your browser and navigate by the path - localhost/< Your_drupal_folder >.

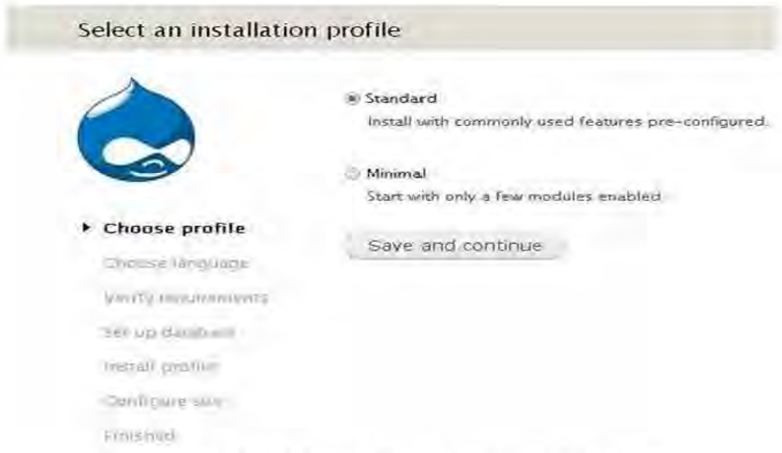


Figure-13.13 installation profile type

Next, select the Standard option and click Save and continue.

Step-4: Next, select the default language for the Drupal website.

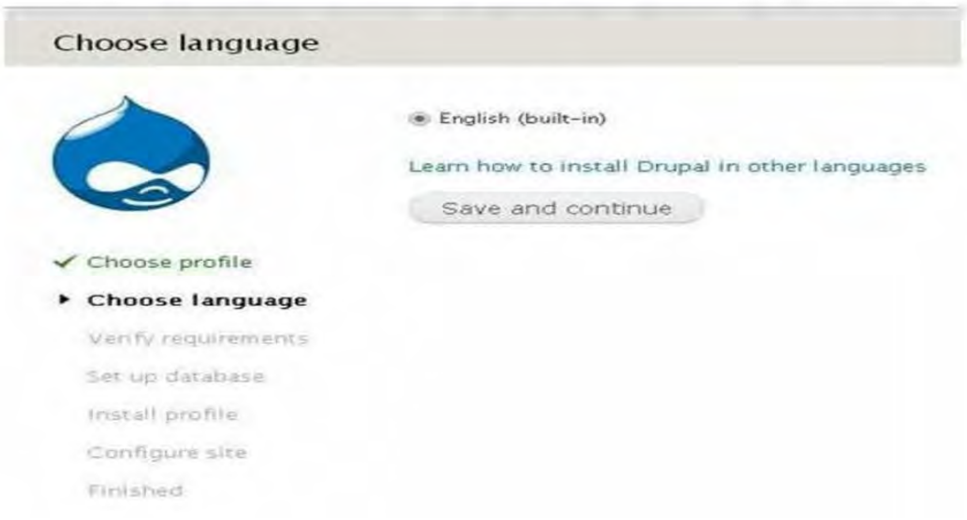
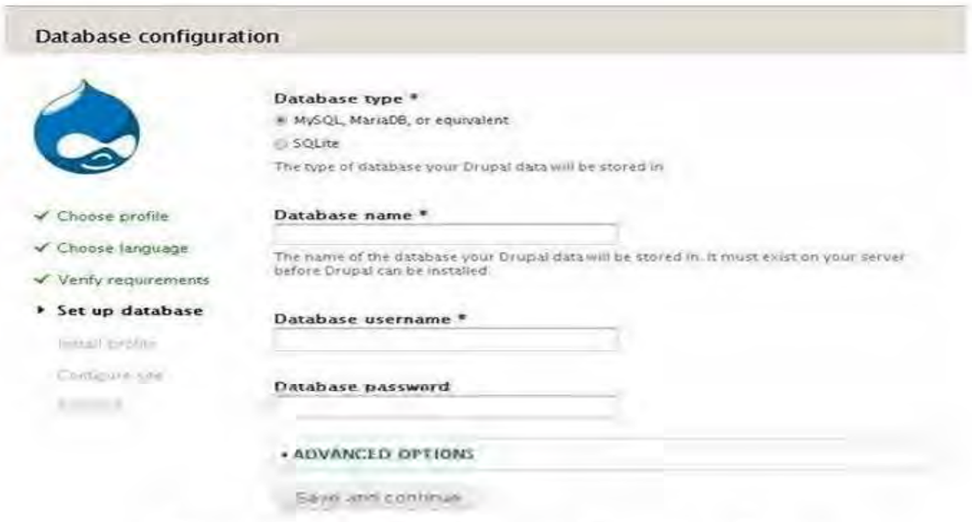


Figure-13.14 language selection

Choose language and click on Save and continue.

Step-5: Next is the Database configuration page, here you need to enter the type of the database that you want to use, Database name, Database username and Database password.


Database configuration

Database type *

[✓ MySQL, MariaDB, or equivalent] SQLite

The type of database your Drupal data will be stored in

Database name *

The name of the database your Drupal data will be stored in. It must exist on your server before Drupal can be installed.

Database username *

Database password

Figure-13.15 Database configuration Page

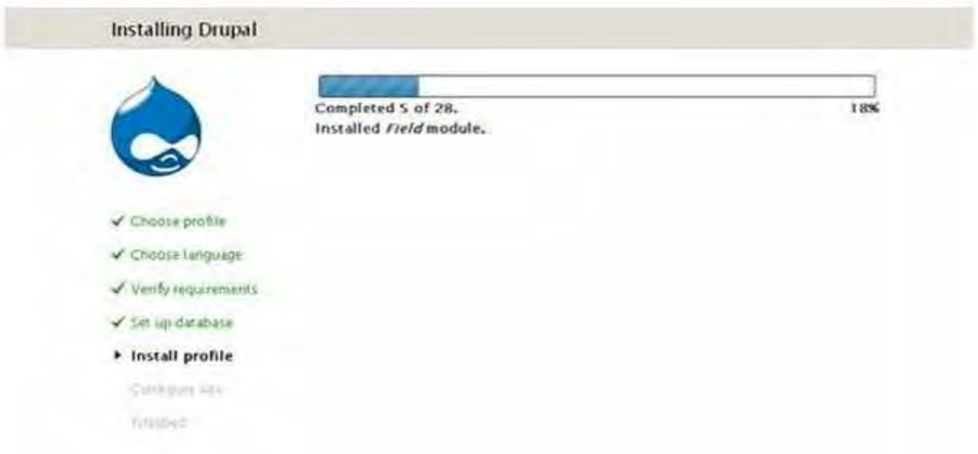
- Database Type: In this category, you have to select the database type. By default it will be MySQL.
- Database name: Here enter the database name for Drupal.
- Database username: Enter the user name of your MySQL database.
- Database password: Enter the password which you had set for MySQL Database.

When you click on the ADVANCED OPTIONS, you will see a screen as shown below:

Here you can fill the advanced options for the database:

- Database Host: Enter the host name. By default it is localhost.
- Database port: Enter the database port. By default it is 80.
- Table Prefix: It is used to add prefix in the database tables which helps to run multiple sites on the same database.
- After filling all information, click on Save and continue button.

Step-6: Next, you will see the installation process starts on your machine.


Installing Drupal

Completed 5 of 28. 18%

Installed *Field* module.

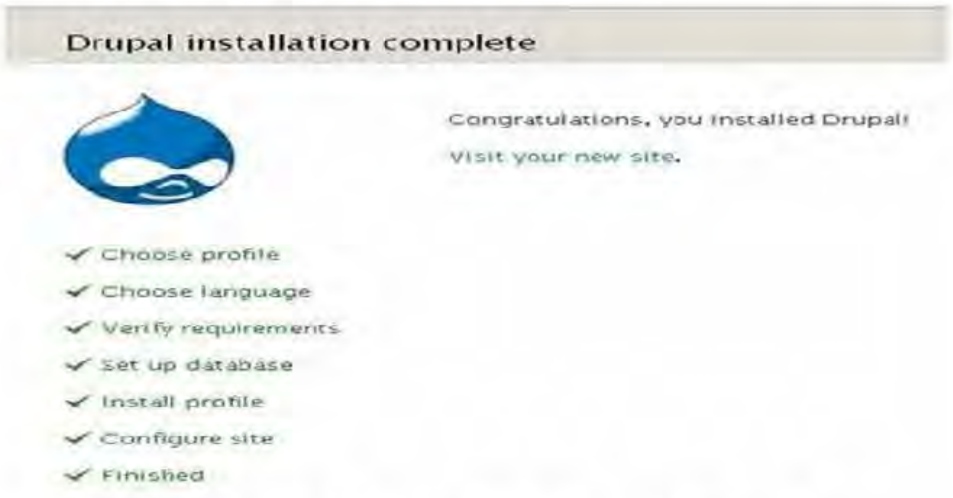
Figure-13.16 Installation Page

Step-7: After that you will get a Configure site page:

It contains the following fields:

- Site name: The name that you want to give to your site. For example: - Vardhman Mahaveer Open University, Kota.
- Site e-mail address: The email address that automated e-mails will be sent from this address.
- Username, Email-address and Password: These are all administrative details used for the maintenance account.
- You need to enter all these fields and click on Save and continue.

Step-8: Next, you will see the below page indicating that your Drupal installation is successful:

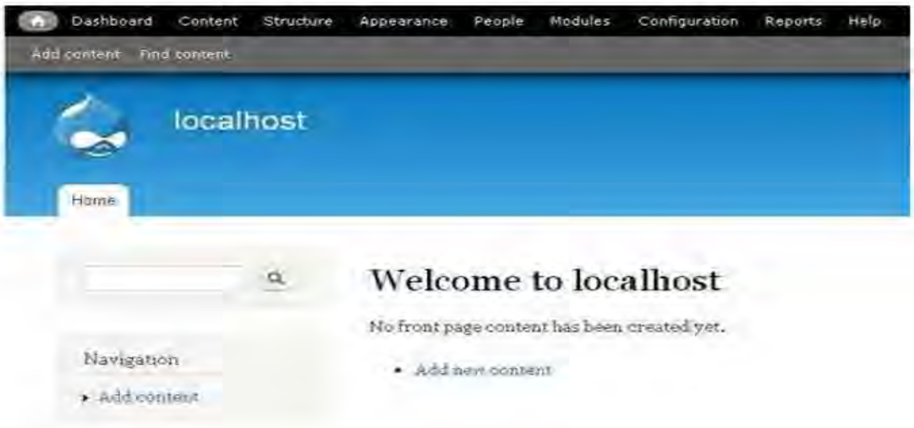

Drupal installation complete

Congratulations, you installed Drupal!

Visit your new site.

Figure-13.17 Drupal installation complete page

Step-9: Click the Visit your new site link as shown in the above image. You will get the newly installed Drupal homepage as shown below.



Welcome to localhost

No front page content has been created yet.

Figure-13.18 Welcome screen

13.3.6 Drupal-Architecture

Drupal is a powerful web content management platform for building simple as well as complex sites. In this section, we will see the architectural style of Drupal for implementing user interfaces. The following diagram shows basic flow of Drupal working:

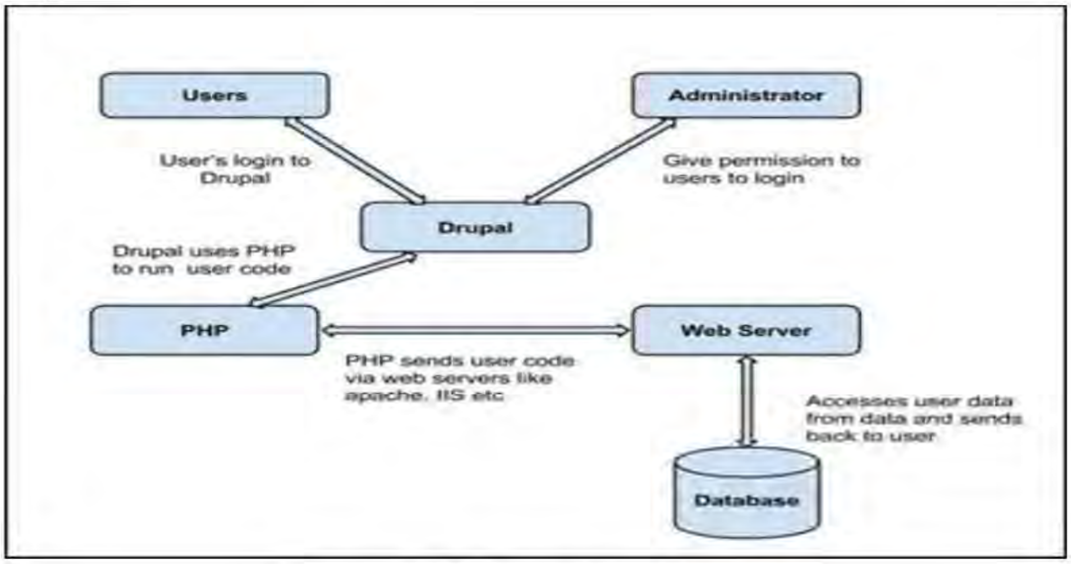

Users
Administrator
User's login to Drupal
Give permission to users to login
Drupal
Drupal uses PHP to run user code
PHP
Web Server
PHP sends user code via web servers like apache, IIS etc
Accesses user data from data and sends back to user
Database

Figure-13.19 Drupal Architecture

The architecture of Drupal contains following components:

- Users
- Administrator
- Drupal

- PHP
- Web Server
- Database

Users: These are Drupal community users. The user sends a request to a server using Drupal CMS and where web browsers, search engines etc., acts like clients.

Administrator: This provides access permission to other users and will be able to block unauthorized access and also managing content/s.

PHP: PHP is server-side scripting language. Drupal is based on PHP and use it in order to work with an application which is created by a user. It takes the services of web server to fetch data from the database. PHP memory requirements depend on the modules which are used in your site. Drupal 6 requires at least 16MB, Drupal 7 requires 32MB and Drupal 8 requires 64MB.

Web Server: Web server is a server where the user interacts and processes requests via HTTP (Hyper Text Transfer Protocol) and serves files that form web pages to web users. The communication between user and server are associated with HTTP. You can use different types of web servers such as apache, IIS, nginx, lighttpd etc.

Database: The database stores the user information, content and many more required data of the site. Drupal uses the database to extract the data from a database and enables to store, modify and update the database.

13.3.7 Drupal Default Module

There are total 44 default modules presented in Drupal site after installation. You can see all these modules in Core section. By default 29 modules are enabled. A list of modules in CORE section can be explored by clicking on Modules.

Following is the list of default modules installed in Drupal 7. These are the core modules required by Drupal and it could not be disabled.

- Block: Controls the constructed page with visual building blocks. Blocks are boxes of content provided into an area or region of a web page.
- Color: Permits administrators to modify the color scheme of compatible themes.

- Comment: Permits users to comment and discuss about the published content.
- Contextual links: Provides contextual links to execute actions associated with elements on a page.
- Dashboard: Offers a dashboard page in the administrative interface for forming administrative projects and tracking information within your website.
- Database logging: Logs and records system events to the database.
- Field: Field API allows adding fields to entities like nodes and users.
- Field SQL storage: It sets to store field data in an SQL database.
- Field UI: User interface for the Field API.
- File: Specifies a file field type.
- Filter: Filters content and exhibits in order.
- Help: Manages the display of online help.
- Image: Allows image handling tools.
- List: Specifies list field types. Create choice lists with this selection.
- Menu: Permits administrators to customize the site navigation menu as per the choice.
- Node: Allows content to be submitted to the site and displayed on pages.
- Number: It sets the numeric field types.
- Options: It specifies the choice, check box and radio button widgets for text and numeric fields.
- Overlay: It specifies the Drupal administration interface in an overlay.
- Path: Permits users to rename URLs.
- RDF: It improves your content with meta data to allow other applications (e.g. search engines, aggregators) understand in a better manner, its relationships and attributes.
- Search: Permits site-wide keyword searching.
- Shortcut: Permits users to manage customizable lists of shortcut links.

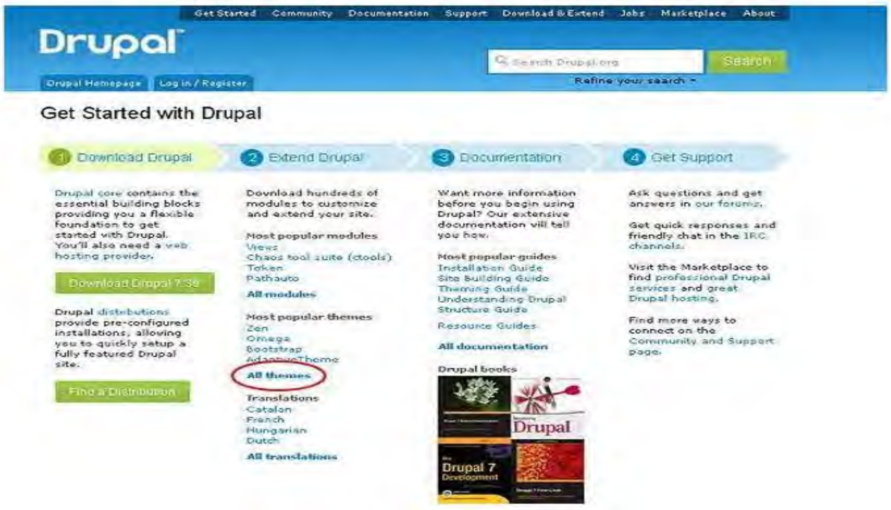
- System: Handles general site configuration for administrators.
- Taxonomy: Enables the categorization of content.
- Text: Defines simple text field types.
- Toolbar: Provides a toolbar that shows the top-level administration menu items and links from other modules.
- Update manager: Checks for available updates and can securely install or update modules and themes via a web interface.
- User: Manages the user registration and login system.

13.3.8 Drupal Theme and Layouts

Theme and Layouts are an important part in Drupal. Bartik theme is the default theme during installation. You can select paid or free themes from Drupal official site. In general, layout is an arrangement of text and graphics. The following process you have to do.

Step-1: Go to Drupal official website.

Step-2: Click on All Themes as shown below.



Get Started with Drupal

1 Download Drupal

Drupal core contains the essential building blocks providing you a flexible foundation to get started with Drupal. You'll also need a web hosting provider.

Download Drupal 7.39

Drupal distributions provide pre-configured installations, allowing you to quickly setup a fully featured Drupal site.

Find a Distribution

2 Extend Drupal

Download hundreds of modules to customize and extend your site.

Most popular modules
Views
Chaos tool suite (ctools)
Token
Pathauto
All modules

Most popular themes
Zen
Omega
Bootstrap
AdaptiveTheme
All themes

Translations
Catalan
French
Hungarian
Dutch
All translations

3 Documentation

Want more information before you begin using Drupal? Our extensive documentation will tell you how.

Most popular guides
Installation Guide
Site Building Guide
Theming Guide
Understanding Drupal
Structure Guide

Resource Guides

All documentation

Drupal books

4 Get Support

Ask questions and get answers in our forums.

Get quick responses and friendly chat in the IRC channels.

Visit the Marketplace to find professional Drupal services and great Drupal hosting.

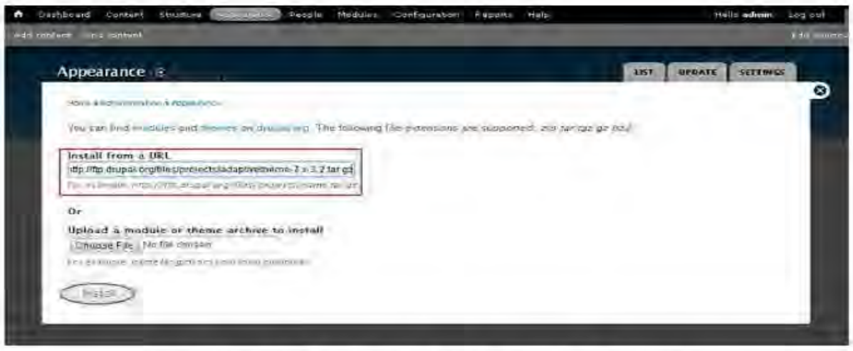
Find more ways to connect on the Community and Support page.

Figure-13.20 Drupal Theme Page

Step-3: Next, you will get a list of themes and select theme of your choice.

Step-4: Download theme file and copy the link. You can also download the theme directly by clicking on archive file and the module will be downloaded locally on your computer.

Step-5: Next, go to Appearance and click install new theme as shown below or paste the link address you copied in step-4 and click on install button as shown below.



You can find modules and themes on drupal.org. The following file extensions are supported: zip tar tgz gz bz2

Figure-13.21 Installation of Theme

Step-6: You can also upload the theme archive, instead of copying the link address, if you have downloaded it. Next, installation process screen will show.

Step-7: Next, click on Enable newly added themes.

Step-8: By default the themes are disabled, to make them enable click Enable and set default button.

Step-9: Next, click on Settings to set the appearance for your site. Here you will see the Layout & General Settings option, in the newly theme consists of Standard layout, Tablet Layout, Small touch layout and Panels & Gpanels.

These options control the display settings for the current Admin theme that is already in use. When your site displays the theme, these settings will be used.

Step-10: Once done with all your configurations, click on save configurations.

For more details i.e. themes, module and updates, consider the Drupal official website.

13.3.9 Self Exercise

1. Write the features of Drupal.
2. Write advantages and disadvantes of Drupal.

3. Write the steps to install the Drupal.
4. What are the basic requirements for Drupal?
5. Describes the Drupal Theme and layout in details.

13.4 Joomla

13.4.0 Introduction

Joomla is also a free and open source Content Management System (CMS), which is used to develop websites and online applications. It has the extendable features which allow separating the user interface into front-end templates and back-end templates (administrator). Joomla uses the Object Oriented Programming, and software design patterns developed in PHP, and MySQL (used for storing the data).

This unit describes the basics of Joomla. By using it, you will get an idea that how to work on Joomla Web Content Management System and will be able to create websites with ease. Joomla is based on Mambo CMS which was developed by an Australian company in 2001 and initially released on August 17, 2005. The official version of Joomla 1.0 was released on 22nd September, 2005.

13.4.1 Features

Joomla has its own powerful built-in features (core features).

- User Manager – This tool helps to manage the user information such as permission to edit, access, publish, create or delete the user, change the password and languages. The main part of the user manager is Authentication.
- Content Manager – It allows to manage the content using WYSIWYG editor to create or edit the content in a very simple way.
- Banner Manager – It is used to add or edit the banners on the website.
- Template Manager – It manages the designs that are used on the website. The templates can be implemented without changing the content structure.

- Media Manager – This tool provides to manage the media files and folder in which you can easily upload, organize and manage your media files into your article editor tool.
- Contact Manager – It allows to add contacts, managing the contact information of the particular users.
- Web Link Manager – This tool provides for user of the site and can be sorted into categories.
- Search – It allows users to search the appropriate information on the site. You can use smart indexing, advanced search options, auto suggest searches to make search efficient.
- Menu Manager – It allows to create menus and menu items and can be managed subsequently. You can put menu in any style and in multiple places.
- RSS – It stands for Really Simple syndication which helps your site contents and RSS files to be automatically updated.

13.4.2 Advantages

Joomla is also a flexible and easy content management system. It allows the user to work on it with many advantages:

- It is an open source platform and freely available for use.
- It has easy installation and setup is very simple even if you're not an advanced user.
- It ensures the safety of data content and doesn't allow anyone to edit the data.
- By default, Joomla is compatible with all browsers.
- Since Joomla is so easy to use, as a web designer or developer, you can quickly build sites for your clients. With minimal instructions to the clients, clients can easily manage their sites on their own.
- It provides simple editing with the content as it uses WYSIWYG editor (What You See Is What You Get)
- The templates are very flexible to use.

- Media files can be uploaded easily in the article editor tool.
- It provides easy menu creation tool.

13.4.3 Disadvantages

In addition to the several advantages, it also has some disadvantages:

- It has some compatibility problem while installing several modules, extensions and plugins simultaneously.
- Most of the Plugins and modules are not free in Joomla.
- Development is too difficult to handle when you want to change the layout.
- Joomla is not much SEO (Search Engine Optimization) friendly.
- It makes website heavy to load and run.

13.4.4 System Requirements for Joomla

- Database – MySQL 5.1 +
- Web Server –
    - ✓ WAMP (Windows)
    - ✓ LAMP (Linux)
    - ✓ XAMP (Multi-platform)
    - ✓ MAMP (Macintosh)
    - ✓ Nginx
    - ✓ Microsoft IIS
- Operating System – Cross-platform
- Browser Support – IE (Internet Explorer 7), Firefox, Google chrome
- SSL (Secure Socket Layer) – A valid security certificate is required for HTTPS
- PHP Compatibility – PHP 5.4+ or PHP 5.3.10+

13.4.5 Installation of Joomla

The installation process requires that you should have Joomla software. For this, you can download the Joomla from its official website- http://www.joomla.org/download.html.

Creating Database

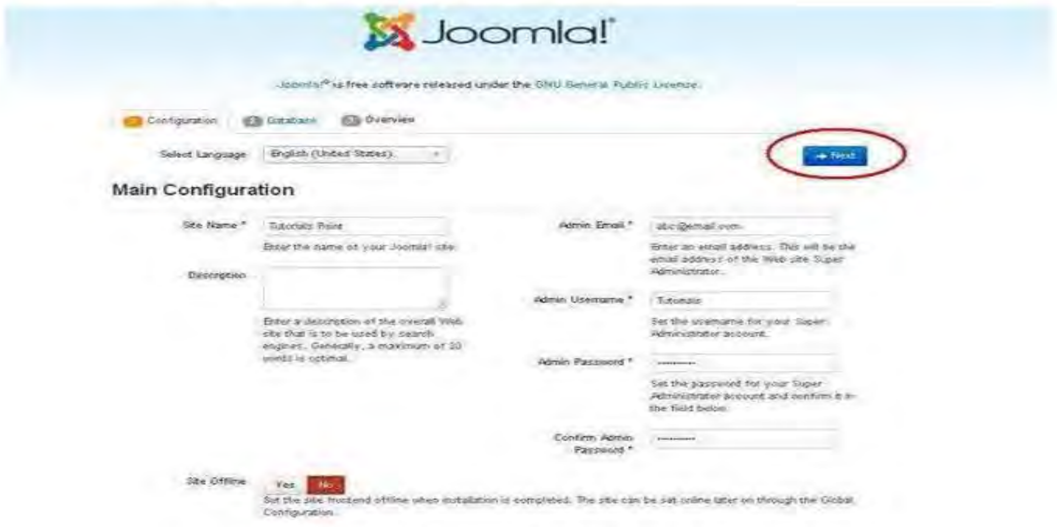
- Joomla requires MySQL database. So create a new empty database and user/password (for e.g. User as "root" and password as "root" or else you can set as per your convenience for Joomla.
- After follows step, you can continue with the installation process.

Installation Set up Wizard

It's very easy to set up Joomla into your system. The following steps are required to setup Joomla locally on your system.

Step-1: Extract the downloaded Joomla folder and upload it on your local web server.

Step-2: Open your browser and navigate to your Joomla file path, then you will get the first screen of the Joomla installer as shown in the following screen. In our case the path is localhost/< Your_joomla_folder >



Joomla!® is free software released under the GNU General Public License.

Configuration | Database | Overview

Select Language: English (United States)

Next

Main Configuration

Site Name *: Tutorials Point
Enter the name of your Joomla! site.

Description:
Enter a description of the overall Web site that is to be used by search engines. Generally, a maximum of 20 words is optimal.

Admin Email *: abc@email.com
Enter an email address. This will be the email address of the Web site Super Administrator.

Admin Username *: Tutorials
Set the username for your Super Administrator account.

Admin Password *: ---------
Set the password for your Super Administrator account and confirm it in the field below.

Confirm Admin Password *: ---------

Site Offline: Yes [✓ No]
Set the site frontend offline when installation is completed. The site can be set online later on through the Global Configuration.

Figure-13.22 Joomla installer screen

It contains the following fields −

- Site Name − Enter the name of the site which you are going to create in Joomla.

- Description – Add a small description about your site.
- Admin Email – Enter your email address which helps to recover our password or any update.
- Admin Username – Enter the username as per your choice while logging into Joomla.
- Admin Password – Enter password to protect your site.
- Site Offline – It specifies whether your site should be offline or online after completion of installation by clicking on Yes/No.

After filling all the information, click on the Next button.

Step-3: Here, you have to enter the information about the MYSQL database. The fields are:

- Database Type – Select your database type. By default it will be MySQLi.
- Host Name – Write the host name by default it is localhost.
- Username – Enter the user name of your MySQL database.
- Password – Enter the password which you had set for MySQL Database.
- Database Name – Enter the database name which you have created in MySQL database for Joomla.
- Table Prefix – It is used to add prefix in the database tables which helps to run multiple sites on the same database. It takes the default value.
- Old Database Process – It gives two options Backup or Remove. If you had already created a database then you can either remove it or select the backup option to create a backup of your whole database information.

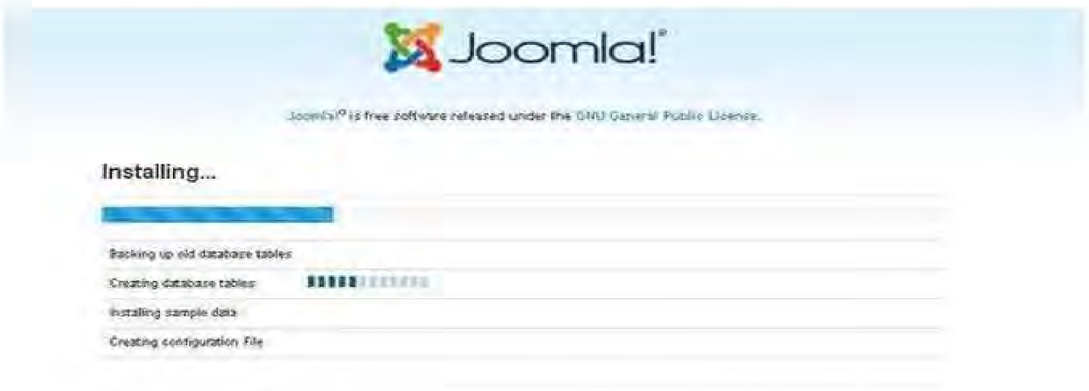
Fill out all required information and click on Next button.

Step-4: In this step, installation process screen will be shown.

Here you can view all the information added to Joomla.

Choose Default English (GB) Sample Data as an example to build your website and click on Install button.

Step-5: Next, you can see that Joomla starts the installation process on your machine.


Joomla!® is free software released under the GNU General Public License.

Installing...

Backing up old database tables
Creating database tables
Installing sample data
Creating configuration File

Figure-13.23 Joomla installation process starts

Step-6: Successful installation of Joomla a congratulation screen will be shown.

Next, click on the "Remove installation folder" button which will help you to protect your site, so that no other person can re-install your site.

After successful installation of Joomla, you are able to login to your Joomla Admin Panel and explore the Joomla shown below –

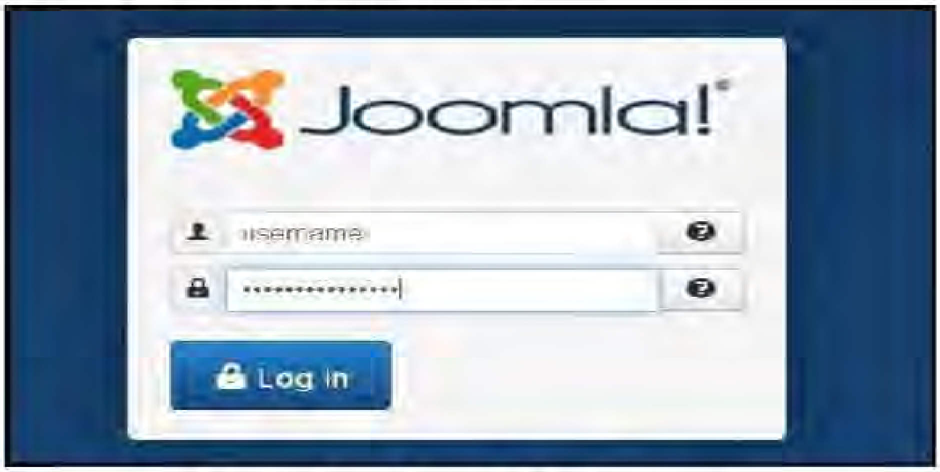


Figure-13.24 Joomla Login screen

13.4.6 Architecture of Joomla

The architecture of Joomla are shown below in the figure and different components are described below–

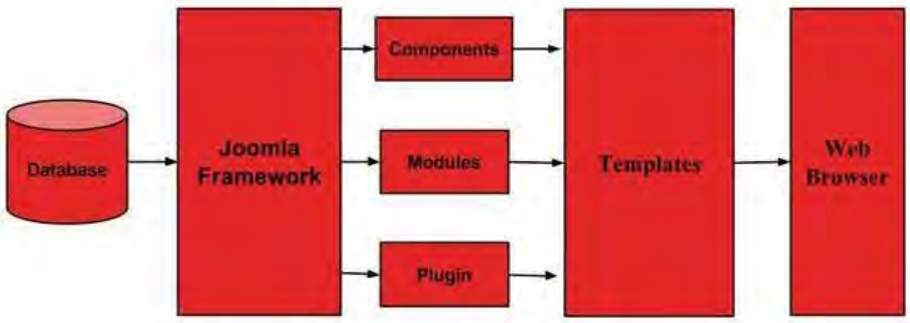


Figure-13.25 Joomla basic architecture

- Database
- Joomla Framework
- Components
- Modules
- Plugin
- Templates
- Web Server

Database – Database is a collection of user data so that we can store, manipulate and organize it. The database stores the user information, content and many more required data of the site. It is used to store the administrative information to manage the Joomla site. This component provides maximum flexibility and compatibility for further extension.

Joomla Framework – It is a collection of open source software, where the Joomla CMS is built. It provides the facility to develop the website into module which helps each package to develop more easily.

Components – Components are main functional unit and acts as small applications. It consists of two parts i.e. Administrator and Site. Whenever a page gets loaded, component is called to render the body of main page. The Administrator part manages different aspects of the component and the site part helps in rendering the pages when request is made by site visitor. of Joomla.

Modules – Modules is an extension which is used to render the pages in Joomla. It is also used to display the new data from the component. It frequently looks like boxes such as login module. In Joomla administrator the modules are managed by the module manager. It displays the new content and images when module is linked to Joomla components.

Plugin – It is a kind of Joomla extension and a very flexible and powerful tool for extending the framework. It contains a bit of codes that is used to execute the particular event trigger. It is commonly used to format the output of a component or module when a page is built.

Templates – Template determines the look and feel of the Joomla website. There are two types of templates used i.e. Front-end and Back-end. The Back-end template is used to control the functions by the administrator where-as the Front-end template is a way to present the website to users. Templates are easy to build or customize your site. It provides maximum flexibility to style your site.

Web Browser – It is a tool where the user interacts. It sends the web pages to the client. The HTTP (Hyper Text Transfer Protocol) is used to communicate between the client and the server.

Control Panel- It provides default features and functions of Joomla to access through clickable icons, menu bar etc. When you login to the Joomla administrative panel, you will get the screen as shown below. Important icons of control panel and their functions are described below–

Article Content

There are four icons under the CONTENT section as shown below –

- Add New Article – It creates a new article page.
- Article Manager – It manages all your present articles.
- Category Manager – It creates new categories and helps in publishing/ un-publishing the categories.
- Media Manager – It manages the files by uploading various new files or deleting the existing ones on your web server.

Structure Format

In STRUCTURE section, there are two groups of icons –

- Menu Manager – Menu manager allows creating custom menus for your website and navigating through your website.

- Module Manager – It manages the modules such as location and function of modules that are installed on site.

User Information

Under USERS section, there is one icon located –

- User Manager – It manages the user information, which allows creating or deleting the user, changing passwords, time and languages. You can also assign the user to User Groups.

Set the configuration

Three sub-components are located under the CONFIGURATION section as given below –

- Global Configuration – This is an important part in the Joomla back-end. Any changes made in this configuration, will affect the entire website.
- Template Manager – It manages the templates used in the website.
- Language Manager – It manages installed language by setting the default language for your site.

Install Extension

There are many Extensions available in Joomla. You can install different types of extensions to extend the functionality of the site.

Maintenance

In MAINTENANCE section, there are two icons located –

- Joomla is up-to-date – It views the current update status of the Joomla installation.
- All extensions are up-to-date – It views the current update status of the Joomla extension.

Logged-In User

It shows the administrator name who has logged in to the Joomla site.

Published Articles

It shows the published articles and also shows the present article, that you have published.

Information of Sites

It displays the details of the site such as OS name, version of PHP and MySQL etc., and also shows the number of users that are using this site.

There are various toolbar options in Joomla. You can explore various toolbars and its sub fields in the Joomla Website –

- Article Manager Toolbar
- Category Manager Toolbar
- Media Manager Toolbar
- Menu Manager Toolbar
- Module Manager Toolbar
- User Manager Toolbar
- Global Configuration Toolbar
- Template Manager Toolbar

For more information about the Joomla web content Management system, you should explore Joomla official website and forum for documentation.

13.5 Summary

In a rapidly changing world, the need for online publishing plays a vital role in needs and expectations of their site visitors. Today, many web publishing system use content management systems (CMS). This permit to rapidly and dynamically update web pages and properties as the new contents becomes available so that every visit to a site is engaging, informative, and meaningful. Every CMS has a different design with its own web functionalities, target to specific users. WordPress, Drupal and Joomla are three world's most popular open source, free, and top quality content management systems. The main purpose of these CMS is to present information on web sites. The functionality of these content management systems is to provide content creation, content management, publishing, presentation and support business goals and strategies.

13.6 Glossary

Software – Software is a set of programs which solve specific or common problem.

Website – It is a collection of webpages.

Open Source - It refers to any program whose source code is made available for use or modification.

Search Engine Optimization - the process of maximizing the number of visitors to a particular website to ensure that site appears high on the list of results returned by a search engine.

W3C compliance – It means that the HTML and CSS code a website has with is fully compliant with the standards set by the World Wide Web Consortium.

13.7 Exercise

Q.1 Briefly explain the Joomla.

Q.2 Write the features of Joomla.

Q.3 Write advantages and disadvantes of Joomla.

Q.4 How to set up the Joomla in our local machine.

Q.5 Compare among the features of Wordpress, Drupal and Joomla

References and Suggested Readings

1. https://wordpress.com/
2. https://wordpress.org/download/
3. https://codex.wordpress.org/
4. https://www.drupal.org/
5. https://www.joomla.org/
6. https://www.tutorialspoint.com