



Government of Nepal  
Ministry of Education, Science and Technology  
Curriculum Development Centre  
Sanothimi, Bhaktapur

Phone : 5639122/6634373/6635046/6630088  
Website : [www.moecdc.gov.np](http://www.moecdc.gov.np)

## Web Development and Database



**Technical and Vocational Stream**  
**Learning Resource Material**

**Web Development and Database**  
**(Grade 12)**

**Secondary Level**  
**Computer Engineering**



Government of Nepal  
Ministry of Education, Science and Technology  
**Curriculum Development Centre**  
Sanothimi, Bhaktapur

**Publisher :** Government of Nepal

Ministry of Education, Science and Technology

**Curriculum Development Centre**

Sanothimi, Bhaktapur

© Publisher

**Layout by Khados Sunuwar**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any other form or by any means for commercial purpose without the prior permission in writing of Curriculum Development Centre.

## **Preface**

The curriculum and curricular materials have been developed and revised on a regular basis with the aim of making education objective-oriented, practical, relevant and job oriented. It is necessary to instill the feelings of nationalism, national integrity and democratic spirit in students and equip them with morality, discipline and self-reliance, creativity and thoughtfulness. It is essential to develop in them the linguistic and mathematical skills, knowledge of science, information and communication technology, environment, health and population and life skills. It is also necessary to bring in them the feeling of preserving and promoting arts and aesthetics, humanistic norms, values and ideals. It has become the need of the present time to make them aware of respect for ethnicity, gender, disabilities, languages, religions, cultures, regional diversity, human rights and social values so as to make them capable of playing the role of responsible citizens with applied technical and vocational knowledge and skills. This Learning Resource Material for Computer Engineering has been developed in line with the Secondary Level Computer Engineering Curriculum with an aim to facilitate the students in their study and learning on the subject by incorporating the recommendations and feedback obtained from various schools, workshops and seminars, interaction programs attended by teachers, students and parents.

In bringing out the learning resource material in this form, the contribution of the Director General of CDC Dr. Lekhnath Poudel, Pro, Dr. Subarna Shakya, Bibha Sthapit, Kumar Prasun, Yogesh Parajuli, Dr. Romakanta Pandey, Rajendra Rokaya, Bimal Thapa, Dinesh Khatri is highly acknowledged. The book is written by Chandra Praksha Timalina and the subject matter of the book was edited by Badrinath Timalina and Khilanath Dhamala. CDC extends sincere thanks to all those who have contributed in developing this book in this form.

This book is a supplementary learning resource material for students and teachers. In addition they have to make use of other relevant materials to ensure all the learning outcomes set in the curriculum. The teachers, students and all other stakeholders are expected to make constructive comments and suggestions to make it a more useful learning resource material.



## Content

	Page No.
Unit 1 : Introduction to HTML	1
Unit 2 : JavaScript	17
Unit 3 : Cascading Style Sheet	40
Unit 4 : Introduction to PHP, MYSQL and APACHE	51
Unit 5 : Configuration of Apache and PHP	55
Unit 6 : Programming Structure of PHP	75
Unit 7 : Working with Form	102
Unit 8 : Conceptof database	109
Unit 9 : MySQL and PHP	124



# Unit: 1

## Introduction to HTML

### Concept of WWW

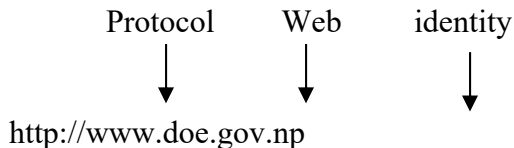
WWW is a way of accessing information over the medium of the internet. WWW is the collection of linked documents or pages stored on millions of computers and distributed across the world. The www is distributed file management system with programs to access and send information. The terms internet and www, which is a service of internet, are many times used interchangeably. However, these are both different; www is a body of information while the internet refers to physical structure of global network. WWW is collection boundless information that uses internet to be accessible worldwide.

### Component of WWW and URL

The URL is uniform resource locator which refers the address of resource in internet.

World

Widecountry



### Domain name

### Introduction to internet

Internet stands for international network. Internet is a network of network (global network of computer). Millions of computers all over the world are connected through the internet. Computer users on the internet can contact with one another anywhere in the world. if computer is connected to the internet, one can connect to millions of computers. in internet, a huge resources of information is accessible to people across the world. Information in every field sharing from education, science, health, medicine, history and geography to business, news etc. can be retrieved



through internet. one can also download program and software packages from internet.

### **History of the internet**

In 1969, the department of defence(DOD) of the united states of America started a project to allow researchers and military personnel to communicate with each other in an emergency. The project was called ARPANET (Advanced Research Project Agency network) and it is the foundation of the internet. To share hardware and software resources, the military allowed universities to join the network, from where the students caught up with it and developed much of the software, so developed the network. later, in 1991, Tim burners-lee developed HTML and formed internet.

### **Protocol**

Protocol is set of rules and language for sending and receiving information over internet. Due to protocol, computer can communicate each other in network/internet. Example of protocol are : TCP/IP,HTTP,SMTP,FTP.

### **Web browser**

A web browser is a software application used to locate into client computer and display web pages. Example of web browser is internet explorer, opera mini, Google chrome, Netscape navigator.

### **Search engine**

A search engine helps user to locate or access the required information from internet in an efficient manner. It is a web site which is used to search the information or website using specific keyword. Examples are

<http://www.goolge.com>

<http://www.askjeevs.com>

<http://www.hotbot.com>

<http://www.infoseek.com>

<http://www.lycos.com>

<http://www.netguide.com>

<http://www.allthewen.com>

<http://www.yahoo.com>

<http://www.ask.com>

<http://www.bing.com>

## **HTML Editors**

An HTML editor is a software tool to create and modify HTML documents. HTML editors can be divided into three categories: Text editors WYSIWYG editors and online editors. Examples of HTML editor are

- Dreamweaver
- Notepad ++
- Edit Plus

## **Basic HTML construct, Building blocks**

HTML is a simple scripting language which is used to create hypertext documents on the World Wide Web. It was invented by Tim Berners Lee in around 1990. It is simply a collection of certain key words called 'Tags' that are helpful in writing the document to be displayed using a browser on internet.

Basic structure of HTML

```
<html>
```

```
<head>
```

```
<title> title of webpage</title>
```

```
</head>
```

```
<body>
```

```
//Body Part// (//This is a comment//)
```

```
</body>
```

```
</html>
```

HTML elements are the building blocks of the HTML web page. The elements

consist of a pair of tags (starting and ending tags) and the textual or graphical content inside of the tags.

## **HTML Tags, Attribute and Value**

An *HTML tag* is commonly defined as a set of characters constituting a formatted command for a Web page. HTML tags are paired tags (closed) and Singular tags (Empty)

### **Paired tags (closed)**

Paired tags is said to be a paired tag if the text is placed between a tag and its companion tag. Inpaired tags, the first tag is referred to as Opening Tag and the second tag is referred to as Closing Tag. Example of paired tags are <html>, <head>, <body>, heading tags(<h1>- <h6>, <font>, <b> etc.

### **Singular tags (Empty)**

Singular tags is said to be empty tag which does not have closing tags. Examples of singular tags are <hr>, <br>, <!-->

Attributes are extra bits of information which can be added to a tag. The purpose of attributes is to add additional style to the element. Values are the number or character to determine the style of the attribute.

Syntax : < tag attribute = "value" >..... </ tag>

Example : <font colour="red">this subject belongs to web and database of computer engineering class 12<font >

Here,

Tag: font

Attribute: colour

Value: red

### **Heading tag**

Heading tag are used to determine the level of heading in website. There are six level of heading <H1> to <H6>

### Example:

```
<H1> this is heading</H1>
```

```
<H2>this is heading</H2>
```

```
<H3>this is heading</H3>
```

```
<H4>this is heading</H4>
```

```
<H5>this is heading</H5>
```

```
<H6>this is heading</H6>
```

### Output

# This is a heading

## This is a heading

### This is a heading

This is a heading

**This is a heading**

**This is a heading**

HTML automatically adds an extra blank line before and after a heading. A useful heading attribute is “align”.

### Paragraph tags

Paragraph tag is used to start the paragraph in webpage. Paragraphs are defined with the <p> tag. Think of a paragraph as a block of text. You can use the align attribute with a paragraph tag as well.

### Example:

```
<p align =”left”>This is a paragraph</p>
```

```
<p align =”center”>This is another paragraph</p>
```

### Output

This is a paragraph

This is another paragraph

## Line Breaks <br>

The tag is used when you want to start a new line, but don't want to start a new paragraph. The tag forces a line break wherever you place it. It is similar to single spacing in a document.

### Example

This <br> is a para<br> graph with line breaks

### Output

This  
is a para  
graph with line breaks

## Font tags

Font tag is used to defining the attribute of font to webpage. The <font> tag defines the font characteristics. Size, colour and type face are defined by the size, colour and face attributes.

### Example:

<font face="arial" colour="red" size=7>This page shows font colour</font>

## List

HTML provides a simple way to show unordered lists (bullet lists) or ordered lists (numbered lists).

**Unordered Lists:** An unordered list is a list of items marked with bullets (typically small black circles). An unordered list starts with the <ul>tag. Each list item starts with the <li>

Tag. Unordered list have some attribute like type. The values for this type attribute are disc, square, circle. By default, circle is available in website.

### Example

```
<ul type="square">  
<li> coffee</li>
```

```
<li>milk</li>
</ul>
```

## Output

- Coffee
- Milk

## Ordered list

An ordered list is also a list of items. The list items are marked with different numbers. An ordered list starts with the `<ol>` tag. Each list item starts with the `<li>` tag. If we do not use type attribute, items are marked with numbers. We use **type = "a"** for lowercase letters list, **type = "I"** for roman numbers list, and **type = "i"** for lowercase numbers list.

```
<ol>
<li>coffee</li>
<li>milk</li>
</ol>
```

## Output

1. Coffee
2. Milk

## HTML Link

HTML uses the anchor tag to create a link to another document or web page. The Anchor Tag and the href Attribute An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc. The syntax of creating an anchor:

```
<a href="url" Text to be displayed</a>
```

The `<a>` tag is used to create an anchor to link from, the href attribute is used to tell the address of the document or page we are linking to, and the words between the open and close of the anchor tag will be displayed as a hyperlink.

There are three type of link used in HTML.

1. Inline: this link is used to link on different area of same page.
2. Internal: this link is used to link from one page to another page.
3. External: this link is used to link from one page to another page in web.
4. We can use **name** attribute to define a named anchor inside a HTML document. Named anchor are invisible to the reader. For example, `<a name = "label">Any content</a>` defines a named anchor and we use the syntax `<a href = "#label">Any content</a>` to link to the named anchor. We can also use named anchor to link to some content within another document. For example,

`<a href="http://www.moecdc.gov.np">Jump to the Useful Tips section</a>`.

#### **For example:**

1. Inline `<a href="#name"> name </a>`
2. Internal `<a href="information.html"> information</a>`
3. External `<a href="www.yahoo.com"> for click yahoo</a>`

#### **Images**

Image is the most important part in website. HTML images are defined with `<img>` tag. To display an image on a page, you need to use the **src** attribute. We can also use **width** and **height** attributes with `img` tag.

For example, `<img src = "photo1.jpg" width = "104" height = "142" />`.

We can use `alt` attribute to define an alternate text for an image. For example, `<img src = "photo1.jpg" width = "104" height = "142" alt = "My best photo"/>`. The "alt" attribute tells the reader what he or she is missing on a page if the browser can't load images. The browser will then display the alternate text instead of the image. It is a good practice to include the "alt" attribute for each image on a page, to improve the display and usefulness of your document for people who have text-only browsers.

#### **For example**

``



Figure: image

## Table

Tables are defined with the `<table>` tag. A table is divided into rows (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag). The letters td stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc. For example,

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr><td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

## Output

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2



We use border attribute to display table with border as shown in the above example. Headings in a table are defined with **<th>**tag. For example,

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

### Output

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

### HTML Forms

Forms are used to select different types of user input. A form is an area that contains different form elements (like text fields, text area fields, drop-down menus, radio buttons, checkboxes etc.). Form elements are elements that allow the user to enter information in a form. A form is defined with the **<form>**tag. For example,

```
<form>
input elements
</form>
```

The most commonly used form tag is **<input>** tag. The type of input is specified with the **type attribute** within the **<input>** tag. For example,

```
<form>
```

```
  First name: <input type="text" name="firstname" />
```

```
<br />
```

```
  Last name: <input type="text" name="lastname" />
```

```
</form>
```

### Output:

**First name:**

**Last name:**

Another input type is **radio button**. Radio buttons are used when you want the user to select one of a limited number of choices. For example,

```
<form>
```

```
<input type="radio" name="sex" value="male" /> Male
```

```
<br />
```

```
<input type="radio" name="sex" value="female" /> Female
```

```
<br/>
```

```
<input type="button" name="subbtn" value="Submit" />
```

```
<input type="button" name="subbtn" value="Reset" />
```

```
</form>
```

### Output:

Male

Female

Another input type is **checkboxes**. Checkboxes are used when you want to select one or more options of a limited number of choices. For example,

```
<form> I have a bike: <input type="checkbox" name="vehicle" value="Bike" />
```

```
<br /> I have a car: <input type="checkbox" name="vehicle" value="Car" />  
<br/>I have an airplane: <input type="checkbox" name="vehicle" value=  
"Airplane" />  
</form>
```

## Output

I have a bike:

I have a car:

I have an airplane:

Another input type is **submit button**. When the user clicks on the "Submit" button, the content of the form is sent to the server. The form's **action attribute** defines the name of the file to send the content to. The file defined in the action attribute usually does something with the received input. For example,

```
<form name="input" action=" submit.php" method="get"> Username: <input  
type="text" name="user" /><input type="submit" value="Submit" /></form>
```

## Output

Username:

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "submit.php". The page will show you the received input.

***Note:** You can use other different form elements as well.*

The **method** attribute of <form> tag specifies how to send form-data (the form-data is sent to the page specified in the action attribute). We can use “**get**” and “**post**” as values of method attribute. When we use get, form-data can be sent as URL variables and when we use post, form-data are sent as HTTP post.

## Notes on the "get" method

- This method appends the form-data to the URL in name/value pairs
- There is a limit to how much data you can place in a URL (varies between browsers), therefore, you cannot be sure that all of the form-data will be

correctly transferred

- Never use the "get" method to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar)

### Notes on the "post" method

- This method sends the form-data as an HTTP post transaction
- The "post" method is more robust and secure than "get", and "post" does not have size limitations

We can create a simple drop-down box on an HTML page. A drop-down box is a selectable list. See code below:

```
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="fiat">Fiat</option>
<option value="audi">Audi</option>
</select>
```

### Output



### HTML Frames

We can use frames to display more than one web page in the same browser window. Each HTML document is called a frame, and each frame is independent of the others. The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page

We use **<frameset>**tag to define how to divide the window into frames. Each frameset defines a set of rows or columns. Within frameset, we use **<frame>**tag to define what HTML document to put into each frame.

If a frame has visible borders, the user can resize it by dragging the border. To

prevent a user from doing this, you can add `noresize="noresize"` to the `<frame>` tag. Add the `<noframes>` tag for browsers that do not support frames.

### **Important**

You cannot use the `<body></body>` tags together with the `<frameset></frameset>` tags. However, if you add a `<noframes>` tag containing some text for browsers that do not support frames, you will have to enclose the text in `<body></body>` tags.

### **Example 1**

```
<frameset cols="25%,50%,25%">
<frame src="frame_a.htm" noresize="noresize"/>
<frame src="frame_b.htm"/>
<frame src="frame_c.htm"/>
<noframes>
<body>Your browser does not handle frames!</body>
</noframes>
</frameset>
```

### **Example 2**

```
<frameset rows="25%,50%,25%">
<frame src="frame_a.htm"/>
<frame src="frame_b.htm"/>
<frame src="frame_c.htm"/>
</frameset>
```

### **Example 3: Mixed Frameset**

```
<frameset rows="50%,50%">
<frame src="frame_a.htm"/>
<frameset cols="25%,75%">
<frame src="frame_b.htm"/>
<frame src="frame_c.htm"/>
</frameset>
</frameset>
```

**Exercise:**

1. How to comment HTML tags?
2. How to create a hyperlink?
3. What is the correct way to write address in an HTML document?
4. How to create an area inside an image-map?
5. Create an HTML document which uses article element?
6. Create an HTML document which uses aside element?
7. How to embed audio in a HTML document?
8. How to write bold text using HTML tags?
9. How to define a section that is quoted from another source?
10. How to define a clickable button?
11. How to define a table caption?
12. How to define a dialog box or window with an HTML tag?
13. How to emphasize text in an HTML document?
14. How to define a footer for a document or section?
15. How to define a label for an input element?

# Unit: 2

## JavaScript

### Overviews of Java Script

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities. JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow

Client-side script to interact with the user and make dynamic pages. It is an Interpreted programming language with object-oriented capabilities. JavaScript is an interpreted, client-side, event-based, object oriented scripting language that you can use to add dynamic interactivity to your web pages.

### Integrate JavaScript within HTML documents

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows:

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

### Example of Script in <head>...</head> section.

```
<html>
<head>
<script type="text/javascript">
document.write("hello world man"+"\\n");
</script>
</head>
```

```
<body>
<h1> saraswati school</h1>
</body>
</html>
```

**Example of Script in <body>...</body> section.**

```
<html>
<body>
<script type="text/javascript">
document.write("hello world");
</script>
</body>
</html>
```

**Example of Script in <body>...</body> and <head>...</head> sections.**

```
<html>
<head>
<script type="text/javascript">
document.write("hello man");
</script></head><body>
<script type="text/javascript">
document.write("hello world");
</script></body></html>
```

**Example of Script in an external file and then include in <head>...</head> section.**

```
<html>
<head>
<body>
<script src="abc.js">
</script>
</body>
```



</html>

## **You need another file with extension css**

abc.js

```
document.write("hello man");
```

## **Variables**

Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. Like many other programming languages, JavaScript has different type of variables.

## **Rules for naming variables are**

- Variable names are case sensitive
- They must begin with a letter or the underscore character.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.

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

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

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

## **Data types in JavaScript**

The data types of a language describe the basic elements that can be used within that language. JavaScript allows you to work with three primitive data types:

- **Numbers**, e.g., 123, 120.50 etc.
- **Strings** of text, e.g. "This text string" etc.
- **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 Number Data types

JavaScript has only one Number (numeric) data types. Number data type can store normal integer, floating-point values.

A floating-point represent a decimal integer with either decimal points or fraction expressed (refer to another decimal number).

```
var num1 = 5;           // Numeric integer value
var num2 = 10.5;       // Numeric float value
var num3 = -30.47;     // Negative numeric float value
var num4 = 5E4;        // 5 x 10 Powers of 4 = 50000
var num5 = 5E-4        // 5 x 10 Powers of -4 = -50000
var num6 = 10 / 0;     // Number divide by zero, result is: Infinity
var num7 = 10 / -0;    // Number divide by negative zero, result is: -Infinity
var num8 = 10, num9 = 12.5; // Multiple variable declare and initialize
```

### JavaScript String Data types

JavaScript string data type represent textual data surrounding to single/double quotes. Each character is represent as a element that occupies the position of that string. Index value 0, starting from first character of the string.

```
var name = 'Hello, I am run this town.!' // Single quote
var name = "Hello, I am run this town.!" // Double quote
```

Whatever quote you use to represent string, but you should take care that quote can't repeat in string statement.

```
var name = "Hello, I'm run this town.!" // Single quote use inside string
var name1 = "" // empty string
```

**Note :** *JavaScript empty string is different from the NULL value.*

## JavaScript Boolean Data types

JavaScript Boolean type can have two value true or false. Boolean type is use to perform logically operator to determine condition/expression is true.

```
var val1 = true;
var val2 = false;
```

## JavaScript Null

JavaScript Null specifies variable is declare but values of this variable is empty.

```
var str = null; // Value assign null
document.writeln(str == undefined); // Returns true
document.writeln(null == undefined); // Equality check Returns true
document.writeln(null === undefined); // Equality with type check Returns false
JavaScript empty variables boolean context return false.
var bool = Boolean();
console.log(bool); // Default Boolean context Returns false
var bool = Boolean(true);
console.log(bool); // Returns true
```

## JavaScript undefined

JavaScript uninitialized variables value are undefined.

Uninitialized variable (value undefined) equal to null. JavaScript uninitialized variables boolean context return false.

```
var str; // Declare variable without value. Identify
as undefined value.
document.writeln(str == null); // Returns true
document.writeln(undefined == null); // Returns true
var bool = Boolean(str); // str is undefined passed into Boolean
object
document.writeln(bool); // Boolean context Returns false
```

## JavaScript comments

JavaScript supports following comments

- 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>
```

## Conditional Statements

Conditional statements are statements which are used to perform different actions for different decisions. You can use conditional statements in your code. We have the following conditional statements in JavaScript.

- If statement
- If...else statement
- If...else if...else statement
- Switch statement

### Example of if statement

```
<script type="text/javascript">
// write "good morning " greeting if the time is less than 10
var d=new Date()
var time=d.getHours()
if (time<10)
{
Document.write("<b> good morning</b>")
}
</script>
```

### Example of if...else statement

```
<script type="text/javascript">
// write "good morning " greeting if the time is less than 10
var d=new Date()
var time=d.getHours()
if (time<10)
{
document.write("<b> good morning</b>")
}
else
{
document.write("<b>good day</b>")
}
</script>
```

### Example of if...else if.....else statement

```
<script type="text/javascript">
// write "good morning " greeting if the time is less than 10
var d=new Date()
```

```

var time=d.getHours()
if (time<=10)
{
document.write("<b> good morning</b>")
}
else if(time>10 && time<16)
{
document.write("<b>good day</b>")
}
else
{
Document.write("<b>good evening</b>")
}
</script>

```

## 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, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

## Example

```

<html>
<body>
<script type = "text/javascript">
<!--

```

```
var count;
document.write("Starting Loop" + "<br />");
for(count = 0; count < 10; count++) {
    document.write("Current Count : " + count );
    document.write("<br />");
}
document.write("Loop stopped!");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

## Output

Starting Loop

Current Count : 0

Current Count : 1

Current Count : 2

Current Count : 3

Current Count : 4

Current Count : 5

Current Count : 6

Current Count : 7

Current Count : 8

Current Count : 9

Loop stopped!

Set the variable to different value and then try...

## The while Loop

The most basic loop in JavaScript is the **while** loop which would be discussed in

this chapter. 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.

### Example

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

### Output

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5

### Function

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions. JavaScript allows us to write our own functions as well. There is some step to use the function as in c programming.



- Function definition
- Function call
- Function return

```

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

```

Whenever click on try it button then, javascript call the function myfunction () and display the “paragraph changed”.

## **Array**

An array is a special variable, which can hold more than one value, at a time. An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own ID so that it can be easily accessed. The following code creates an Array object called myCars:

```
var myCars=new Array();
```

## **Accessing the Array**

You can refer to a particular element in an array by referring to the name of the

array and the index number. The index number starts at 0. In above initialized array, the code line `document.write(myCars[0]);` will result in the following output: Saab  
To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

 will result in the following output: Opel.

### Some methods associated with array

- **concat() :** Joins two or more arrays, and returns a copy of the joined arrays
- **join() :** Joins all elements of an array into a string
- **pop() :** Removes the last element of an array, and returns that element
- **push() :** Adds new elements to the end of an array, and returns the new length
- **reverse() :** Reverses the order of the elements in an array
- **shift() :** Removes the first element of an array, and returns that element
- **sort() :** Sorts the elements of an array
- **toString() :** Converts an array to a string, and returns the result
- **unshift() :** Adds new elements to the beginning of an array, and returns the new length

### Example

#### Concat() : Joining Two Arrays

```
<script type="text/javascript"> var parents = ["Giri", "Pari"];  
var children = ["Cactus", "Rose"]; var family = parents.concat(children);  
document.write(family); </script>
```

The output will be :

Giri, Pari, Cactus, Rose

### String Object in JavaScript

The String object is used to manipulate a stored piece of text. String objects are created with new String().

## Syntax

var txt = new String(string);or more simply:

var txt = string;

## Some methods associated with String object:

- **toLowerCase( )**: Converts a string to lowercase letters
- **toUpperCase( )**: Converts a string to uppercase letters
- **concat( )**: Joins two or more strings, and returns a copy of the joined strings
- **charAt( )**: Returns the character at the specified index
- **indexOf( )**: Returns the position of the first found occurrence of a specified value in a string
- **replace( )**: Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring

## Examples

In the following example we are using the length property of the String object to return the number of characters in a string:

```
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.length);
</script>
```

The output of the code above will be: 12

In the following example we are using the toUpperCase( ) method of the String object to display a text in uppercase letters:

```
<script type="text/javascript">
var str="hello its me webtech!";
document.write(str.toUpperCase());
</script>
```

The output of the code above will be:

**HELLO ITS ME WEBTECH**

## Example: IndexOf ( ) method

The index Of( ) method returns the position of the first occurrence of a specified value in a string. This method returns -1 if the value to search for never occurs. The indexOf( ) method is case sensitive.

### Syntax

*string*.indexOf(searchstring, start)

**searchstring:** Required. The string to search for.

**start:** Optional. The start position in the string to start the search. If omitted, the search starts from position 0

```
<script type="text/javascript">
var str="Patan world!";
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("WORLD") + "<br />");
document.write(str.indexOf("world"));
</script>
```

### Output

10 -16

## Math Object in Javascript

The Math object allows you to perform mathematical tasks. The Math object includes several mathematical constants and methods. For example

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

**Note:** Math is not a constructor. All properties and methods of Math can be called by using Math as an object without creating it.

### Properties

- **Math.E:** Returns Euler's number (approx. 2.718)
- **Math.LN2:** Returns the natural logarithm of 2 (approx. 0.693)
- **Math.LN10:** Returns the natural logarithm of 10 (approx. 2.302)

- **Math.LOG2E:** Returns the base-2 logarithm of E (approx. 1.442)
- **Math.LOG10E:** Returns the base-10 logarithm of E (approx. 0.434)
- **Math.PI:** Returns PI (approx. 3.14159)
- **Math.SQRT1\_2:** Returns the square root of 1/2 (approx. 0.707)
- **Math.SQRT2:** Returns the square root of 2 (approx. 1.414)

## Methods

- **abs(x):** Returns the absolute value of x
- **ceil(x):** Returns x, rounded upwards to the nearest integer
- **floor(x):** Returns x, rounded downwards to the nearest integer
- **log(x):** Returns the natural logarithm (base E) of x
- **max(x,y,z,...,n):** Returns the number with the highest value
- **min(x,y,z,...,n):** Returns the number with the lowest value
- **pow(x,y):** Returns the value of x to the power of y
- **sqrt(x):** Returns the square root of x
- **random():** Returns a random number between 0 and 1
- **round(x):** Rounds x to the nearest integer
- **sin(x):** Returns the sine of x (x is in radians)
- **cos(x):** Returns the cosine of x (x is in radians)
- **tan(x):** Returns the tangent of an angle

## Examples

```
document.write(Math.round(4.7));
```

Output: 5

```
document.write(Math.random());
```

Output: 0.19733826867061233

```
document.write(Math.floor(Math.random()*6));
```

**Output: 3**

## Date Object in Javascript

The Date object is used to work with dates and times. Date objects are created with the Date( ) constructor. We can easily manipulate the date by using the methods

available for the Date object. In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();  
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date(); myDate.setDate(myDate.getDate()+5);
```

**Note:** If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

## Methods

- getDate() Returns the day of the month (from 1-31)
- getDay() Returns the day of the week (from 0-6)
- getFullYear() Returns the year (four digits)
- getHours() Returns the hour (from 0-23)
- getMilliseconds() Returns the milliseconds (from 0-999)
- getMinutes() Returns the minutes (from 0-59)
- getMonth() Returns the month (from 0-11)
- getSeconds() Returns the seconds (from 0-59)
- setDate() Sets the day of the month (from 1-31)
- setFullYear() Sets the year (four digits)
- setHours() Sets the hour (from 0-23)
- setMilliseconds() Sets the milliseconds (from 0-999)
- setMinutes() Set the minutes (from 0-59)
- setMonth() Sets the month (from 0-11)
- setSeconds() Sets the seconds (from 0-59)
- toString() Converts a Date object to a string

## Examples

The Date object is also used to compare two dates. The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();
```

```
myDate.setFullYear(2010,0,14);  
var today = new Date(); if (myDate>today) { alert("Today is before 15th December  
2011"); } Else { alert("Today is after 15th January 2011"); }
```

## Examples

```
<html>  
<head>  
<script type="text/javascript">  
function displayDate()  
{  
document.getElementById("demo").innerHTML=Date();  
}  
</script>  
</head>  
<body>  
<h1>My First Web Page</h1>  
<p id="demo">This is a paragraph.</p>  
<button type="button" onclick="displayDate()">Display Date</button>  
</body>  
</html>  
  
<html>  
<body>  
<script type="text/javascript">  
var d=new Date();  
document.write(d);  
</script>  
</body>  
</html>
```

## Example: Displaying the clock

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
//m=checkTime(m);
//s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',1000);
}
//to concat 0 if i is not double digit
/*function checkTime(i)
{
if (i<10)
{
i="0" + i;
}
return i;
} */
</script>
</head>
<body onload="startTime()">
```



```
<div id="txt"></div>
</body>
</html>
```

With JavaScript, it is possible to execute some code after a specified time-interval. This is called timing events. It's very easy to time events in JavaScript. The two key methods that are used are:

- `setTimeout()` - executes a code some time in the future
- `clearTimeout()` - cancels the `setTimeout()`

**Note:** The `setTimeout()` and `clearTimeout()` are both methods of the HTML DOM Window object.

The `setTimeout()` method returns a value. In the syntax defined above, the value is stored in a variable called `t`. If you want to cancel the `setTimeout()` function, you can refer to it using the variable name. The first parameter of `setTimeout()` can be a string of executable code, or a call to a function. The second parameter indicates how many milliseconds from now you want to execute the first parameter.

**Note:** There are 1000 milliseconds in one second.

In above example the function `startTime()` get executed after each second, showing the content of `div` tag getting refreshed each time so as to display the clock.

### **User defined objects in JavaScript:**

We have seen that JavaScript has several built-in objects, like `String`, `Date`, `Array`, and more. In addition to these built-in objects, you can also create your own.

An object is just a special kind of data, with a collection of properties and methods.

Let's illustrate with an example: A person is an object. Properties are the values associated with the object. The persons' properties include name, height, weight, age, skin tone, eye color, etc. All persons have these properties, but the values of those properties will differ from person to person. Objects also have methods. Methods are the actions that can be performed on objects. The persons' methods could be `eat()`, `sleep()`, `work()`, `play()`, etc.

**The syntax for accessing a property of an object is:**

```
objName.propName
```

**You can call a method with the following syntax:**

```
objName.methodName()
```

**Note:** Parameters required for the method can be passed between the parentheses.

There are different ways to create a new object:

### **Create a direct instance of an object**

The following code creates a new instance of an object, and adds four properties to it:

```
personObj=new Object();
personObj.firstname="Jyoti";
personObj.lastname="Joshi";
personObj.age=25;
personObj.eyecolor="black";
```

alternative syntax (using object literals):

```
personObj={firstname:"Jyoti", lastname:"Joshi", age:25, eyecolor:"black"};
```

Adding a method to the personObj is also simple.

The following code adds a method called eat() to the personObj:

```
personObj.eat=eat;
function eat( )
{
// code for the function
}
```

### **Event**

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc. Some common examples of event are

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key

### Onclick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type.

```
<html>
<head>
<script type = "text/javascript">
<!--
    function sayHello() {
        alert("Hello World")
    }
//-->
</script>
</head>
<body>
<p>Click the following button and see result</p>
<form>
<input type = "button" onclick = "sayHello()" value = "Say Hello" />
</form>
</body>
</html>
```

## Output

Click the following button and see result

### Onmouseover and Onmouseout

These two event types will help you create nice effects with images or even with text as well. The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element.

```
<html>
<head>
<script type = "text/javascript">
<!--
    function over() {
        document.write ("Mouse Over");
    }
    function out() {
        document.write ("Mouse Out");
    }
    //-->
</script>
</head>
<body>
<p>Bring your mouse inside the division to see the result:</p>
<div onmouseover = "over()" onmouseout = "out()">
<h2> This is inside the division </h2>
</div>
</body>
</html>
```

## Output

Mouse Over

### How to get data from Form

To obtain a reference to a text input element, here is some sample code:

```
<!DOCTYPE html>
<html>
<body>
<form action="/action_page.php" method="post">
<input type="text" name="fname" required>
<input type="submit" value="Submit">
</form>
<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>
</body>
</html>
```

### Concept of JQUERY

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation. The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

## Exercise

1. Write a JavaScript program to display the current day and time in the following format. *Sample Output* : Today is : Tuesday. Current time is : 10 PM : 30 : 38
2. Write a JavaScript program to find the area of a triangle where lengths of the three of its sides are 5, 6, 7.
3. Write a JavaScript program to determine whether a given year is a leap year in the Gregorian calendar.
4. Write a JavaScript program to calculate multiplication and division of two numbers (input from user).

## Sample form

1st Number :

2nd Number:

The Result Is :  
120

## Unit: 3

# Cascading Style Sheet

### Webpage and CSS

CSS stands for cascading style sheets. It was first developed in 1997, as a way for Web developers to define the look and feel of their web pages. It was intended to allow developers to separate content from design and layout so that HTML could perform more of the function without worry about the design and layout. It is used to separate style from content.

CSS has a *selector* and one or more *declarations*. Selector is normally the HTML element you want to style and each declaration consists of a *property* and *value*. The property is the style attribute we want to use and each property has a value associated with it.

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

### Basic Elements of CSS Design

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts

**Selector:** A selector is an HTML tag at which a style will be applied. This could be any tag like `<h1>` or `<table>` etc.

- **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border*, etc.
- **Value:** Values are assigned to properties. For example, *color* property can have the value either *red* or *#F1F1F1* etc.

### Example:

```
p {color:red;text-align:center;}
```

CSS implementation

We can use style sheets in three different ways in HTML document. They are :

- **External style sheet**
- **Internal style sheet**
- **Inline stylesheet**

### **External Style Sheet**

If we want to apply the same style to many pages, we use external style sheet. With an external style sheet, you can change the look of an entire Web site by changing one style sheet file. Each page must link to the style sheet using the **<link>** tag. The **<link>** tag goes inside the head section. For example,

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a **.css** extension. An example of a style sheet file is shown below:

```
hr {color:sienna;}
p {margin-left:20px;} /*Note: Do not leave space between property value and
units*/
body {background-image:url("abc.jpg");}
```

### **Internal Style Sheet**

If you want a unique style to a single document, an internal style sheet should be used. You define internal styles in the head section of an HTML page, by using the **<style>** tag. For example,

```
<head>
<style type="text/css">
hr {color:red;}
p {margin-left:20px;}
```



```
body {background-image:url("images/b.jpg");}
</style>
</head>
```

## Inline Styles

If you want a unique style to a single element, an inline style sheet should be used. An inline style loses many of the advantages of style sheets by mixing content with presentation. To use inline styles you use the **style attribute** in the relevant tag. The style attribute can contain any CSS property. For example,

```
<p style="color:yellow;margin-left:20px">This is a paragraph.</p>
```

## Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers. A CSS comment begins with "/\*", and ends with "\*/".

## HTML, Id and Class Selectors

The selector which uses HTML tag as selector is called HTML selector. The **id** selector is used to specify a style for a single, unique element. The id selector uses id attribute of the HTML element and is defined with "#". For example,

```
<head>
<style type="text/css">
#para1
{
text-align:center;
color:red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
```

```
<p>This paragraph is not affected by the style.</p>
</body>
```

The **class** selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for any HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a ".". For example,

```
<head>
<style type="text/css">
.center
{
text-align:center;
}
</style>
</head>
<body>
<h1 class="center">Center-aligned heading</h1>
<p class="center">Center-aligned paragraph.</p>
</body>
```

You can also specify that only specific HTML elements should be affected by a class. For example,

```
<head>
<style type="text/css">
p.center
{
text-align:center;
}
</style>
</head>
<body>
```

```
<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be center-aligned.</p>
</body>
```

## Multiple Styles Will Cascade into One

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

**Tip:** Even multiple external style sheets can be referenced inside a single HTML document.

## CSS Background

Inline

```
<h1 style="background-color:red">abc</h1>
```

### internal

```
body{background-color:red}
```

### External

```
<head>
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
<p>This is background </p>
<H1>This is thick</H1>
<H2>This is thick</H2>
<H3>This is thick</H3>
<H4>This is thick</H4>
</body>
</html>
```

## Style.css

```
body{  
background: blue  
}
```

## CSS Text properties

The CSS font properties define the font family, boldness, size, and the style of a text. The font-size property sets the size of the text. The font-style property is mostly used to specify italic text. The font family of a text is set with the font-family property.

```
<html>  
<head>  
<style>  
h1 {color:black; font-size: 40px;}  
h2 {color:red;font-family:century;font-style:italic}  
h3 {color:blue}  
</style>  
</head>  
<body>  
<h1>Ministry of Education science and Technlogy</h1>  
<h2>Nepal</h2>  
<h3>Technical and Vocational Education </h3>  
</body>  
</html>
```

## Background and Images CSS

The background-color property specifies the background color of an element.

```
<!DOCTYPE html>  
<html>  
<head>
```

```

<style>
body {
  background-color: blue;
}
img {
  border: 1px solid #ddd;
  border-radius: 4px;
  padding: 5px;
  width: 150px;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>This page has a light blue background color!</p>

<h2> Images</h2>
<p>Use the border property to create thumbnail images:</p>
</body>
</html>

```

## Lists

The list-style-type property specifies the type of list item marker.

```

<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
  list-style-type: circle;

```

```

}
ul.b {
  list-style-type: square;
}
ol.c {
  list-style-type: upper-roman;
}
ol.d {
  list-style-type: lower-alpha;
}
</style>
</head>
<body>
<p>Example of unordered lists:</p>
<ul class="a">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ul>
<ul class="b">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
<ol class="c">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>

```

```
</ol>
<ol class="d">
<li>Coffee</li>
<li>Tea</li>
<li>Coca Cola</li>
</ol>
</body>
</html>
```

## Links

With CSS, links can be styled in different ways. In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

## Example

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}
/* visited link */
a:visited {
  color: green;
```

```

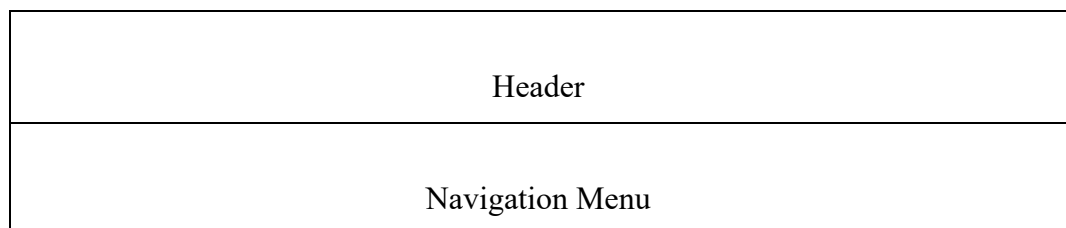
}

/* mouse over link */
a:hover {
  color: hotpink;
}
/* selected link */
a:active {
  color: blue;
}
</style>
</head>
<body>
<p><b><a href="default.html" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS
definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order
to be effective.</p>
</body>
</html>

```

### Basic design of layout

A website is often divided into headers, menus, content and footer. There are different layout use for designing the website. However, the structure below is one of the most common.





Content	Main content	Content
Footer		

Figure: layout of webpage

### Exercise

1. What is CSS? What are advantages of using CSS?
2. What are the three main ways to add CSS to a webpage? Describe the advantages and disadvantages of each method.
3. List the basic layout components of the CSS box model with a brief description for each.
4. Describe the following common CSS units of length: cm, em, in, mm, pc, pt, and px.
5. How do you define a pseudo class in CSS? What are they used for?

## **Unit: 4**

### **Introduction to PHP, MYSQL and APACHE**

#### **Introduction of Static and Dynamic Website**

##### **Static Website**

Static website is a most basic type of website which is easiest to create. These websites do not require any Web programming (like PHP, ASP.NET) or database design (like MySQL). A static site can be built by simply creating a few HTML pages and publishing them to a web server. Since static websites contain fixed code, the content of each page does not change unless it is manually updated. Visitor can't change content of website (Eg: login registration will not be there).

##### **Dynamic Website**

A dynamic website is a site that contains dynamic pages such as templates, contents, scripts etc. The dynamic website displays various content types every time it is browsed. The web page can be changed with the reader that opens the page, character of consumer interplay, or day time. A dynamic site is one that is written using a server-side scripting language such as PHP, ASP, JSP etc.

##### **Server-Side Scripting Language**

Server-side scripting is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page. Server-side scripts provide an interface to the user and are used to limit access to proprietary data and help keep control of the script source code.

Many languages may be used to create these scripts. They include but are not limited to the examples below:

PHP, ASP, Java, JavaScript (using SSJS (Server-side JavaScript) e.g., node.js), Perl, Python, R, Ruby etc.

## Web Server

A web server is server software/hardware dedicated to running said software, that can satisfy World Wide Web client requests. A web server can, in general, contain one or more websites. A web server processes incoming network requests over HTTP and several other related protocols. The basic function of a web server is to host websites and to deliver web content from its hosted websites over the internet. During the delivery of web pages, web servers follow a network protocol known as hypertext transfer protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to the text content.

### Process on web server

When the URL of a website is processed from browser, the browser connects to the server and sends a request for the page. After the complete processing, the servers send back the requested page. Finally, the user is able to view the page on the computer screen. So, basically there are three processes involved in the complete scenario.

1. Establishing a connection
2. Sending a request
3. Responding to the request

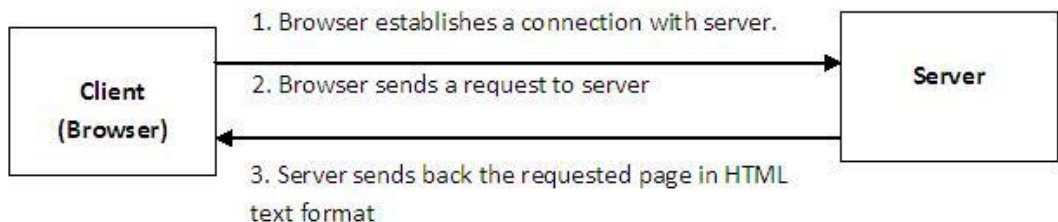


Figure : Process of web server

### Introduction of Apache

Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can

be highly customized to meet the needs of many different environments by using extensions and modules. Most WordPress hosting providers use Apache as their web server software. However, WordPress can run on other web server software as well.

## **PHP**

Hypertext Pre-processor is a general-purpose programming language originally designed for web development. PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. PHP is a widely-used, free, and efficient. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group.

The major difference between HTML and PHP is that, HTML is a language used to describe to a browser how to display text and other objects in a browser window. It is not a programming language. HTML works on a client computer (the system on which the page is being viewed). PHP is a scripting language, and can be used to create dynamic web pages written in HTML.

## **MySQL**

MySQL is an open-source relational database management system. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

## **Introduction to WAMP**

WAMP is sometimes used as an abbreviated name for the software stack Windows, Apache, MySQL, PHP. It is derived from LAMP which stands for Linux, Apache, MySQL, and PHP. As the name implies, while LAMP is used on Linux servers, WAMP is used on Windows servers.

## **Installing Apache, PHP, MySQL**

There are many software which covers the all these software like Apache, Mysql and PHP. WAMP is such type of software. WAMP is a variation of LAMP for Windows systems and is often installed as a software bundle (Apache, MySQL, and PHP). It is often used for web development and internal testing, but may also be

used to serve live websites. Step by step for the installation

1. Download the WAMP Server

I'll give you simple steps for 'Installation of WAMP Server'. We need to download the WAMP server Package .exe file from official WAMP server page. Choose file with Config of your system.

Start Setup Wizard

2. You'll get below Setup Wizard window, When you Run WAMP server .exe file. It shows config of WAMP. Click 'Next' button to continue.

You'll get below Setup Wizard window, When you Run WAMP server .exe file. It shows config of WAMP. Click 'Next' button to continue.

1. Follow the instruction
2. Finally click on ok

## Unit: 5

### Configuration of Apache and PHP

Apache server has a very powerful, but slightly complex, configuration system of its own. After installing the WAMP software on your computer, start to configure the software. Once WAMP Server is installed, it's time to configure the server environment as per your requirements. Single left-click on the WAMP Server icon shows a menu— **localhost, phpMyAdmin, www directory, Apache, PHP, MySQL, webGrind, Start All Services, Stop All Services, Restart All Services, and Put Online/Offline.**

You can also access

<http://localhost/?phpinfo=1>

and

<http://localhost/phpmyadmin/>

on your browser. phpMyAdmin is used to manage MySQL databases and related operations.

#### **php.ini basic (localhost://php.ini)**

The PHP configuration file, `php.ini`, is the final and most immediate way to affect PHP's functionality. The `php.ini` file is read each time PHP is initialized. In other words, whenever `httpd` is restarted for the module version or with each script execution for the CGI version. If your change isn't showing up, remember to stop and restart `httpd`. If it still isn't showing up, use `phpinfo()` to check the path to `php.ini`.

Installation is the first step in making Apache functional. Before you begin, you should have a clear idea of the installation's purpose. This idea, together with your paranoia level, will determine the steps you will take to complete the process. The system-hardening matrix presents one formal way of determining the steps. Though every additional step you make now makes the installation more secure, it also increases the time you will spend maintaining security. As a rule of thumb, if you are building a high profile web server—public or not—always go for a highly

secure installation.

Though the purpose of this chapter is to be a comprehensive guide to Apache installation and configuration, you are encouraged to read others' approaches to Apache hardening as well. Every approach has its unique points, reflecting the personality of its authors. Besides, the opinions presented here are heavily influenced by the work of others. The Apache reference documentation is a resource you will go back to often. In addition to it, ensure you read the Apache Benchmark, which is a well-documented reference installation procedure that allows security to be quantified. It includes a semi-automated scoring tool to be used for assessment.

The following is a list of some of the most useful Apache installation documentation encountered:

- Apache Online Documentation (<http://httpd.apache.org/docs-2.0/>)
- Apache Security Tips ([http://httpd.apache.org/docs-2.0/misc/security\\_tips.html](http://httpd.apache.org/docs-2.0/misc/security_tips.html))
- Apache Benchmark (<http://cisecurity.org/en-us/?route=downloads.browse.category.benchmarks.servers.web.apache>)
- “Securing Apache: Step-by-Step” by Artur Maj (<http://www.securityfocus.com/print/infocus/1694>)
- “Securing Apache 2: Step-by-Step” by Artur Maj (<http://www.securityfocus.com/print/infocus/1786>)

## **Installation**

The installation instructions, given in this chapter, are designed to apply to both active branches (1.x and 2.x) of the Apache web server running on Linux systems. If you are running some other flavor of Unix, I trust you will understand what the minimal differences between Linux and your system are. The configuration advice given in this chapter works well for non-Unix platforms (e.g., Windows) but the differences in the installation steps are more noticeable:

- Windows does not offer the chroot functionality or an equivalent.
- You are unlikely to install Apache on Windows from source code. Instead,

download the binaries from the main Apache web site.

- Disk paths are different though the meaning is the same.

### **Source or Binary**

One of the first decisions you will make is whether to compile the server from the source or use a binary package. This is a good example of the dilemma mentioned at the beginning of this chapter. There is no one correct decision for everyone or one correct decision for you alone. Consider some pros and cons of the different approaches:

- By compiling from source, you are in the position to control everything. You can choose the compile-time options and the modules, and you can make changes to the source code. *This process will consume a lot of your time*, especially if you measure the time over the lifetime of the installation (it is the only correct way to measure time) and if you intend to use modules with frequent releases (e.g., PHP).
- Installation and upgrade is a breeze when binary distributions are used now that many vendors have tools to have operating systems updated automatically. You exchange some control over the installation in return for not having to do everything yourself. However, this choice means you will have to wait for security patches or for the latest version of your favorite module. In fact, the latest version of Apache or your favorite module may never come since most vendors choose to use one version in a distribution and only issue patches to that version to fix potential problems. This is a standard practice, which vendors use to produce stable distributions.
- The Apache version you intend to use will affect your decision. For example, nothing much happens in the 1.x branch, but frequent releases (with significant improvements) occur in the 2.x branch. Some operating system vendors have moved on to the 2.x branch, yet others remain faithful to the proven and trusted 1.x branch.

### **Note**

The Apache web server is a victim of its own success. The web server from the 1.x branch works so well that many of its users have no need to upgrade. In the long



term this situation only slows down progress because developers spend their time maintaining the 1.x branch instead of adding new features to the 2.x branch. Whenever you can, use Apache 2!

This book shows the approach of compiling from the source code since that approach gives us the most power and the flexibility to change things according to our taste. To download the source code, go to <http://httpd.apache.org> and pick the latest release of the branch you want to use.

### **Downloading the source code**

Habitually checking the integrity of archives you download from the Internet is a good idea. The Apache distribution system works through mirrors. Someone may decide to compromise a mirror and replace the genuine archive with a trojaned version (a version that feels like the original but is modified in some way, for example, programmed to allow the attacker unlimited access to the web server). You will go through a lot of trouble to secure your Apache installation, and it would be a shame to start with a compromised version.

If you take a closer look at the Apache download page, you will discover that though archive links point to mirrors, archive signature links always point to the main Apache web site.

One way to check the integrity is to calculate the MD5 sum of the archive and to compare it with the sum in the signature file. An MD5 sum is an example of a hash function, also known as one-way encryption. The basic idea is that, given data (such as a binary file), a hash function produces seemingly random output. However, the output is always the same when the input is the same, and it is not possible to reconstruct the input given the output. In the example below, the first command calculates the MD5 sum of the archive that was downloaded, and the second command downloads and displays the contents of the MD5 sum from the main Apache web site. You can see the sums are identical, which means the archive is genuine:

```
$ md5sum httpd-2.0.50.tar.gz
8b251767212aebf41a13128bb70c0b41 httpd-2.0.50.tar.gz
```

```
$ wget -O - -q http://www.apache.org/dist/httpd/httpd-2.0.50.tar.gz.md5
8b251767212aebf41a13128bb70c0b41 httpd-2.0.50.tar.gz
```

Using MD5 sums to verify archive integrity can be circumvented if an intruder compromises the main distribution site. He will be able to replace the archives and the signature files, making the changes undetectable.

A more robust, but also a more complex approach is to use *public-key cryptography* for integrity validation. In this approach, Apache developers use their cryptographic keys to sign the distribution digitally. This can be done with the help of GnuPG, which is installed on most Unix systems by default. First, download the PGP signature for the appropriate archive, such as in this example:  

```
$ wget http://www.apache.org/dist/httpd/httpd-2.0.50.tar.gz.asc
```

Attempting to verify the signature at this point will result in GnuPG complaining about not having the appropriate key to verify the signature:

```
$ gpg httpd-2.0.50.tar.gz.asc
```

```
gpg: Signature made Tue 29 Jun 2004 01:14:14 AM BST using DSA key ID
DE885DD3
```

```
gpg: Can't check signature: public key not found
```

GnuPG gives out the unique key ID (DE885DD3), which can be used to fetch the key from one of the key servers (for example, `pgpkeys.mit.edu`):

```
$ gpg --keyserver pgpkeys.mit.edu --recv-key DE885DD3
```

```
gpg: /home/ivanr/.gnupg/trustdb.gpg: trustdb created
```

```
gpg: key DE885DD3: public key "Sander Striker <striker@apache.org>" imported
```

```
gpg: Total number processed: 1
```

```
gpg:          imported: 1
```

This time, an attempt to check the signature gives satisfactory results:

```
$ gpg httpd-2.0.50.tar.gz.asc
```

```
gpg: Signature made Tue 29 Jun 2004 01:14:14 AM BST using DSA key ID
DE885DD3
```

```
gpg: Good signature from "Sander Striker <striker@apache.org>"
```

```
gpg:      aka "Sander Striker <striker@striker.nl>"
gpg:      aka "Sander Striker <striker@striker.nl>"
gpg:      aka "Sander Striker <striker@apache.org>"
```

gpg: checking the trustdb

gpg: no ultimately trusted keys found

Primary key fingerprint: 4C1E ADAD B4EF 5007 579C 919C 6635 B6C0 DE88  
5DD3

At this point, we can be confident the archive is genuine. On the Apache web site, a file contains the public keys of all Apache developers (<http://www.apache.org/dist/httpd/KEYS>). You can use it to import all their keys at once but I prefer to download keys from a third-party key server. You should ignore the suspicious looking message (“no ultimately trusted keys found”) for the time being. It is related to the concept of *web of trust*.

### **Downloading patches**

Sometimes, the best version of Apache is not contained in the most recent version archive. When a serious bug or a security problem is discovered, Apache developers will fix it quickly. But getting a new revision of the software release takes time because of the additional full testing overhead required. Sometimes, a problem is not considered serious enough to warrant an early next release. In such cases, source code patches are made available for download at <http://www.apache.org/dist/httpd/patches/>. Therefore, the complete source code download procedure consists of downloading the latest official release followed by a check for and possible download of optional patches.

### **Static Binary or Dynamic Modules**

The next big decision is whether to create a single static binary, or to compile Apache to use dynamically loadable modules. Again, the tradeoff is whether to spend more time in order to get more security.

- Static binary is reportedly faster. If you want to squeeze the last bit of performance out of your server, choose this option. But, as hardware is

becoming faster and faster, the differences between the two versions will no longer make a difference.

- A static server binary cannot have a precompiled dynamic module *backdoor* added to it. (If you are unfamiliar with the concept of backdoors, see the sidebar [Apache Backdoors](#).) Adding a backdoor to a dynamically compiled server is as simple as including a module into the configuration file. To add a backdoor to a statically compiled server, the attacker has to recompile the whole server from scratch.
- With a statically linked binary, you will have to reconfigure and recompile the server every time you want to change a single module.
- The static version may use more memory depending on the operating system used. One of the points of having a dynamic library is to allow the operating system to load the library once and reuse it among active processes. Code that is part of a statically compiled binary cannot be shared in this way. Some operating systems, however, have a memory usage reduction feature, which is triggered when a new process is created by duplication of an existing process (known as *forking*). This feature, called *copy-on-write*, allows the operating system to share the memory in spite of being statically compiled. The only time the memory will be duplicated is when one of the processes attempts to change it. Linux and FreeBSD support copy-on-write, while Solaris reportedly does not.

## **Downloading Apache for Windows**

Information on the latest versions of Apache can be found on the web site of the Apache web server at <http://httpd.apache.org/download.cgi>. There you will find the current release, as well as more recent alpha or beta test versions, and a list of HTTP and FTP mirrors from which you can download the Apache web server. Please use a mirror near to you for a fast and reliable download.

For Windows installations you should download the version of Apache for Windows with the `.msi` extension. This is a single Microsoft Installer file, which contains a ready-to-run build of Apache. There is a separate `.zip` file, which contains only the source code, see the summary above.

## Installing Apache for Windows

You need Microsoft Installer 2.0 or above for the installation to work. For Windows NT 4.0 and 2000 refer to Microsoft's article [KB 292539](#). Windows XP and later do not require this update. The Windows 98/ME installer engine appears to no longer be available from Microsoft, and these instructions no longer detail such prerequisites.

Note that you cannot install two versions of Apache 2.2 on the same computer with the binary installer. You can, however, install a version of the 1.3 series **and** a version of the 2.2 series on the same computer without problems. If you need to have two different 2.2 versions on the same computer, you have to compile and install Apache from the source.

Run the Apache .msi file you downloaded above. The installation will ask you for these things:

- 1. Network Domain.** Enter the DNS domain in which your server is or will be registered in. For example, if your server's full DNS name is `server.mydomain.net`, you would type `mydomain.net` here.
- 2. Server Name.** Your server's full DNS name. From the example above, you would type `server.mydomain.net` here.
- 3. Administrator's Email Address.** Enter the server administrator's or webmaster's email address here. This address will be displayed along with error messages to the client by default.
- 4. For whom to install Apache** Select `for All Users`, on `Port 80`, as a `Service - Recommended` if you'd like your new Apache to listen at port 80 for incoming traffic. It will run as a service (that is, Apache will run even if no one is logged in on the server at the moment) Select `only for the Current User`, on `Port 8080`, when started `Manually` if you'd like to install Apache for your personal experimenting or if you already have another WWW server running on port 80.
- 5. The installation type.** Select `Typical` for everything except the source code and libraries for module development. With `Custom` you can specify what to

install. A full install will require about 13 megabytes of free disk space. This does *not* include the size of your web site(s).

- 6. Where to install.** The default path is C:\Program Files\Apache Software Foundation under which a directory called Apache2.2 will be created by default.

During the installation, Apache will configure the files in the `conf` subdirectory to reflect the chosen installation directory. However, if any of the configuration files in this directory already exist, they will not be overwritten. Instead, the new copy of the corresponding file will be left with the extension `.default`. So, for example, if `conf\httpd.conf` already exists, it will be renamed as `conf\httpd.conf.default`. After the installation you should manually check to see what new settings are in the `.default` file, and if necessary, update your existing configuration file.

Also, if you already have a file called `htdocs\index.html`, it will not be overwritten (and no `index.html.default` will be installed either). This means it should be safe to install Apache over an existing installation, although you would have to stop the existing running server before doing the installation, and then start the new one after the installation is finished.

After installing Apache, you must edit the configuration files in the `conf` subdirectory as required. These files will be configured during the installation so that Apache is ready to be run from the directory it was installed into, with the documents server from the subdirectory `htdocs`. There are lots of other options which you should set before you really start using Apache. However, to get started quickly, the files should work as installed.

## Customizing Apache for Windows

Apache is configured by the files in the `conf` subdirectory. These are the same files used to configure the Unix version, but there are a few different directives for Apache on Windows. See the [directive index](#) for all the available directives.

The main differences in Apache for Windows are:

- Because Apache for Windows is multithreaded, it does not use a separate

process for each request, as Apache can on Unix. Instead there are usually only two Apache processes running: a parent process, and a child which handles the requests. Within the child process each request is handled by a separate thread.

The process management directives are also different:

MaxRequestsPerChild: Like the Unix directive, this controls how many requests (actually, connections) which a single child process will serve before exiting. However, unlike on Unix, a replacement process is not instantly available. Use the default `MaxRequestsPerChild 0`, unless instructed to change the behavior to overcome a memory leak in third party modules or in-process applications.

**Warning: The server configuration file is reread when a new child process is started. If you have modified `httpd.conf`, the new child may not start or you may receive unexpected results.**

ThreadsPerChild: This directive is new. It tells the server how many threads it should use. This is the maximum number of connections the server can handle at once, so be sure to set this number high enough for your site if you get a lot of hits. The recommended default is `ThreadsPerChild 150`, but this must be adjusted to reflect the greatest anticipated number of simultaneous connections to accept.

- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache may interpret backslashes as an "escape character" sequence, you should consistently use forward slashes in path names, not backslashes. Drive letters can be used; if omitted, the drive of the `SystemRoot` directive (or `-d` command line option) becomes the default.
- While filenames are generally case-insensitive on Windows, URLs are still treated internally as case-sensitive before they are mapped to the filesystem. For example, the `<Location>`, `Alias`, and `ProxyPass` directives all use case-sensitive arguments. For this reason, it is particularly important to use the `<Directory>` directive when attempting to limit access to content in the filesystem, since this directive applies to any content in a directory, regardless

of how it is accessed. If you wish to assure that only lowercase is used in URLs, you can use something like:

```
RewriteEngine On
RewriteMap lowercase int:tolower
RewriteCond %{REQUEST_URI} [A-Z]
RewriteRule (.*) ${lowercase:$1} [R,L]
```

- When running, Apache needs write access only to the logs directory and any configured cache directory tree. Due to the issue of case insensitive and short 8.3 format names, Apache must validate all path names given. This means that each directory which Apache evaluates, from the drive root up to the directory leaf, must have read, list and traverse directory permissions. If Apache2.2 is installed at C:\Program Files, then the root directory, Program Files and Apache2.2 must all be visible to Apache.
- Apache for Windows contains the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the \Apache2.2\modules directory. To activate these or other modules, the new LoadModule directive must be used. For example, to activate the status module, use the following (in addition to the status-activating directives in access.conf):

```
LoadModule status_module modules/mod_status.so
```

Information on creating loadable modules is also available.
- Apache can also load ISAPI (Internet Server Application Programming Interface) extensions such as those used by Microsoft IIS and other Windows servers. Note that Apache **cannot** load ISAPI Filters, and ISAPI Handlers with some Microsoft feature extensions will not work.
- When running CGI scripts, the method Apache uses to find the interpreter for the script is configurable using the ScriptInterpreterSource directive.
- Since it is often difficult to manage files with names like .htaccess in Windows, you may find it useful to change the name of this per-directory configuration file using the AccessFilename directive.
- Any errors during Apache startup are logged into the Windows event log



when running on Windows NT. This mechanism acts as a backup for those situations where Apache is not yet prepared to use the `error.log` file. You can review the Windows Applicat Event Log by using the Event Viewer, e.g. Start - Settings - Control Panel - Administrative Tools - Event Viewer.

## Running Apache as a Service

You can install Apache as a service automatically during the installation. If you chose to install for all users, the installation will create an Apache service for you. If you specify to install for yourself only, you can manually register Apache as a service after the installation. You have to be a member of the Administrators group for the service installation to succeed.

Apache comes with a utility called the Apache Service Monitor. With it you can see and manage the state of all installed Apache services on any machine on your network. To be able to manage an Apache service with the monitor, you have to first install the service (either automatically via the installation or manually).

You can install Apache as a Windows NT service as follows from the command prompt at the Apache `bin` subdirectory:

```
httpd.exe -k install
```

If you need to specify the name of the service you want to install, use the following command. You have to do this if you have several different service installations of Apache on your computer.

```
httpd.exe -k install -n "MyServiceName"
```

If you need to have specifically named configuration files for different services, you must use this:

```
httpd.exe -k install -n "MyServiceName" -f "c:\files\my.conf"
```

If you use the first command without any special parameters except `-k install`, the service will be called `Apache2.2` and the configuration will be assumed to be `conf\httpd.conf`.

Removing an Apache service is easy. Just use:

```
httpd.exe -k uninstall
```

The specific Apache service to be uninstalled can be specified by using:

```
httpd.exe -k uninstall -n "MyServiceName"
```

Normal starting, restarting and shutting down of an Apache service is usually done via the Apache Service Monitor, by using commands like `NET START Apache2.2` and `NET STOP Apache2.2` or via normal Windows service management. Before starting Apache as a service by any means, you should test the service's configuration file by using:

```
httpd.exe -n "MyServiceName" -t
```

You can control an Apache service by its command line switches, too. To start an installed Apache service you'll use this:

```
httpd.exe -k start
```

To stop an Apache service via the command line switches, use this:

```
httpd.exe -k stop
```

or

```
httpd.exe -k shutdown
```

You can also restart a running service and force it to reread its configuration file by using:

```
httpd.exe -k restart
```

By default, all Apache services are registered to run as the system user (the `LocalSystem` account). The `LocalSystem` account has no privileges to your network via any Windows-secured mechanism, including the file system, named pipes, DCOM, or secure RPC. It has, however, wide privileges locally.

**Never grant any network privileges to the `LocalSystem` account! If you need Apache to be able to access network resources, create a separate account for Apache as noted below.**

It is recommended that users create a separate account for running Apache service(s). If you have to access network resources via Apache, this is required.

1. Create a normal domain user account, and be sure to memorize its password.

2. Grant the newly-created user a privilege of Log on as a service and Act as part of the operating system. On Windows NT 4.0 these privileges are granted via User Manager for Domains, but on Windows 2000 and XP you probably want to use Group Policy for propagating these settings. You can also manually set these via the Local Security Policy MMC snap-in.
3. Confirm that the created account is a member of the Users group.
4. Grant the account read and execute (RX) rights to all document and script folders (`htdocs` and `cgi-bin` for example).
5. Grant the account change (RWXD) rights to the Apache `logs` directory.
6. Grant the account read and execute (RX) rights to the `httpd.exe` binary executable.

It is usually a good practice to grant the user the Apache service runs as read and execute (RX) access to the whole Apache2.2 directory, except the `logs` subdirectory, where the user has to have at least change (RWXD) rights.

If you allow the account to log in as a user and as a service, then you can log on with that account and test that the account has the privileges to execute the scripts, read the web pages, and that you can start Apache in a console window. If this works, and you have followed the steps above, Apache should execute as a service with no problems.

**Error code 2186** is a good indication that you need to review the "Log On As" configuration for the service, since Apache cannot access a required network resource. Also, pay close attention to the privileges of the user Apache is configured to run as.

When starting Apache as a service you may encounter an error message from the Windows Service Control Manager. For example, if you try to start Apache by using the Services applet in the Windows Control Panel, you may get the following message:

```
Could not start the Apache2.2 service on \\COMPUTER
Error 1067; The process terminated unexpectedly.
```

You will get this generic error if there is any problem with starting the Apache

service. In order to see what is really causing the problem you should follow the instructions for Running Apache for Windows from the Command Prompt.

If you are having problems with the service, it is suggested you follow the instructions below to try starting httpd.exe from a console window, and work out the errors before struggling to start it as a service again.

### **Running Apache as a Console Application**

Running Apache as a service is usually the recommended way to use it, but it is sometimes easier to work from the command line (on Windows 9x running Apache from the command line is the recommended way due to the lack of reliable service support.)

To run Apache from the command line as a console application, use the following command:

```
httpd.exe
```

Apache will execute, and will remain running until it is stopped by pressing Control-C.

You can also run Apache via the shortcut Start Apache in Console placed to Start Menu --> Programs --> Apache HTTP Server 2.2.xx --> Control Apache Server during the installation. This will open a console window and start Apache inside it. If you don't have Apache installed as a service, the window will remain visible until you stop Apache by pressing Control-C in the console window where Apache is running in. The server will exit in a few seconds. However, if you do have Apache installed as a service, the shortcut starts the service. If the Apache service is running already, the shortcut doesn't do anything.

You can tell a running Apache to stop by opening another console window and entering:

```
httpd.exe -k shutdown
```

This should be preferred over pressing Control-C because this lets Apache end any current operations and clean up gracefully.

You can also tell Apache to restart. This forces it to reread the configuration file.

Any operations in progress are allowed to complete without interruption. To restart Apache, either press Control-Break in the console window you used for starting Apache, or enter

```
httpd.exe -k restart
```

in any other console window.

Note for people familiar with the Unix version of Apache: these commands provide a Windows equivalent to `kill -TERM pid` and `kill -USR1 pid`. The command line option used, `-k`, was chosen as a reminder of the `kill` command used on Unix.

If the Apache console window closes immediately or unexpectedly after startup, open the Command Prompt from the Start Menu --> Programs. Change to the folder to which you installed Apache, type the command `httpd.exe`, and read the error message. Then change to the logs folder, and review the `error.log` file for configuration mistakes. If you accepted the defaults when you installed Apache, the commands would be:

```
c:  
cd "\Program Files\Apache Software Foundation\Apache2.2\bin"  
httpd.exe
```

Then wait for Apache to stop, or press Control-C. Then enter the following:

```
cd ..\logs  
more < error.log
```

When working with Apache it is important to know how it will find the configuration file. You can specify a configuration file on the command line in two ways:

- `-f` specifies an absolute or relative path to a particular configuration file:

```
httpd.exe -f "c:\my server files\anotherconfig.conf"
```

or

```
httpd.exe -f files\anotherconfig.conf
```

- `-n` specifies the installed Apache service whose configuration file is to be used:

```
httpd.exe -n "MyServiceName"
```

In both of these cases, the proper ServerRoot should be set in the configuration file.

If you don't specify a configuration file with `-f` or `-n`, Apache will use the file name compiled into the server, such as `conf\httpd.conf`. This built-in path is relative to the installation directory. You can verify the compiled file name from a value labelled as `SERVER_CONFIG_FILE` when invoking Apache with the `-v` switch, like this:

```
httpd.exe -v
```

Apache will then try to determine its ServerRoot by trying the following, in this order:

1. A ServerRoot directive via the `-C` command line switch.
2. The `-d` switch on the command line.
3. Current working directory.
4. A registry entry which was created if you did a binary installation.
5. The server root compiled into the server. This is `/apache` by default, you can verify it by using `httpd.exe -v` and looking for a value labelled as `HTTPD_ROOT`.

During the installation, a version-specific registry key is created in the Windows registry. The location of this key depends on the type of the installation. If you chose to install Apache for all users, the key is located under the `HKEY_LOCAL_MACHINE` hive, like this (the version numbers will of course vary between different versions of Apache):

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Software  
Foundation\Apache\2.2.2
```

Correspondingly, if you chose to install Apache for the current user only, the key is located under the `HKEY_CURRENT_USER` hive, the contents of which are dependent of the user currently logged on:

```
HKEY_CURRENT_USER\SOFTWARE\Apache Software Foundation\Apache\2.2.2
```

This key is compiled into the server and can enable you to test new versions without affecting the current version. Of course, you must take care not to install the new version in the same directory as another version.

If you did not do a binary install, Apache will in some scenarios complain about the missing registry key. This warning can be ignored if the server was otherwise able to find its configuration file.

The value of this key is the ServerRoot directory which contains the `conf` subdirectory. When Apache starts it reads the `httpd.conf` file from that directory. If this file contains a ServerRoot directive which contains a different directory from the one obtained from the registry key above, Apache will forget the registry key and use the directory from the configuration file. If you copy the Apache directory or configuration files to a new location it is vital that you update the ServerRoot directive in the `httpd.conf` file to reflect the new location.

## Testing the Installation

After starting Apache (either in a console window or as a service) it will be listening on port 80 (unless you changed the Listen directive in the configuration files or installed Apache only for the current user). To connect to the server and access the default page, launch a browser and enter this URL:

```
http://localhost/
```

Apache should respond with a welcome page and you should see "It Works!". If nothing happens or you get an error, look in the `error.log` file in the `logs` subdirectory. If your host is not connected to the net, or if you have serious problems with your DNS (Domain Name Service) configuration, you may have to use this URL:

```
http://127.0.0.1/
```

If you happen to be running Apache on an alternate port, you need to explicitly put that in the URL:

```
http://127.0.0.1:8080/
```

Once your basic installation is working, you should configure it properly by editing

the files in the `conf` subdirectory. Again, if you change the configuration of the Windows NT service for Apache, first attempt to start it from the command line to make sure that the service starts with no errors.

Because Apache **cannot** share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache. These conflicting services include other WWW servers, some firewall implementations, and even some client applications (such as Skype) which will use port 80 to attempt to bypass firewall issues.

### **Configuring Access to Network Resources**

Access to files over the network can be specified using two mechanisms provided by Windows:

Mapped drive letters

e.g., `Alias /images/ Z:/`

UNC paths

e.g., `Alias /images/ //imagehost/www/images/`

Mapped drive letters allow the administrator to maintain the mapping to a specific machine and path outside of the Apache `httpd` configuration. However, these mappings are associated only with interactive sessions and are not directly available to Apache `httpd` when it is started as a service. **Use only UNC paths for network resources in `httpd.conf`** so that the resources can be accessed consistently regardless of how Apache `httpd` is started. (Arcane and error prone procedures may work around the restriction on mapped drive letters, but this is not recommended.)

### **Example DocumentRoot with UNC path**

```
DocumentRoot //dochost/www/html/
```

### **Example DocumentRoot with IP address in UNC path**

```
DocumentRoot //192.168.1.50/docs/
```

### **Example Alias and corresponding Directory with UNC path**

```
Alias /images/ //imagehost/www/images/
```



```
<Directory//imagehost/www/images/>
```

```
...
```

```
<Directory>
```

When running Apache httpd as a service, you must create a separate account in order to access network resources, as described above.

## Basic PHP Scripts

```
<?php echo
```

```
Code ...
```

```
?>
```

## Example

```
<html>
```

```
<head>
```

```
<title>PHP Test</title>
```

```
</head>
```

```
<body>
```

```
<?php echo '<p>Hello World</p>'; ?>
```

```
</body>
```

```
</html>
```

## Unit: 6

### Programming Structure of PHP

#### Variables

The main way to store information in the PHP program is by using a variable. Variables can store data of different types, and different data types can do different things.

Basic rule to define variable in PHP program are as follow:

- All variables in PHP are denoted with a leading dollar sign (\$).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

#### DataTypes in PHP

PHP has a total of eight data types which we use to construct variables:

- **Integers:** are whole numbers, without a decimal point, like 4305.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

The first five are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

## Integers

Integer are whole numbers, without a decimal point, like 4305. These are the simplest type. Integer correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so:

```
<?php
    $int_var = 12345;
    $next_int = -12345 + 12345;
?>
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is  $(2^{31}-1)$  (or 2,147,483,647), and the smallest (most negative) integer is  $(-2^{31})$  (or -2,147,483,647).

## Double

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code:

```
<?php
    $many = 2.2888800;
    $many_2 = 2.211200;
    $few = $many + $many_2;
print(.$many + $many_2 = $few<br>.);
?>
```

It produces the following browser output:

2.28888 + 2.21112 = 4.5

## Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so:

```
<?php
    if (TRUE)
        print("This will always print<br>");
    else
        print("This will never print<br>");
?>
```

### Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type:

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
<?php
    $true_num = 3 + 0.14159;
    $true_str = "Tried and true"
```

```
$true_array[49] = "An array element";  
$false_array = array();  
$false_null = NULL;  
$false_num = 999 - 999;  
$false_str = "";
```

?>

## NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this:

```
<?php
```

```
    $my_var = NULL;
```

```
?>
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed:

```
<?php
```

```
    $my_var = null;
```

```
?>
```

A variable that has been assigned NULL has the following properties:

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

## Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string:

```
<?php
```

```
    $string_1 = "This is a string in double quotes";
```

```
    $string_2 = "This is a somewhat longer, singly quoted string";
```

```
$string_39 = "This string has thirty-nine characters";  
$string_0 = ""; // a string with zero characters
```

?>

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php  
$variable = "name";  
$literally = 'My $variable will not print!\n';  
print($literally);  
$literally = "My $variable will print!\n";  
print($literally);  
?>
```

This will produce the following result:

```
My $variable will not print!\n  
My name will print
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with \$) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character

- \t is replaced by the tab character
- \\$ is replaced by the dollar sign itself (\$)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

## Variable Naming

Rules for naming a variable are:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like +, -, %, (, ), . &, etc.
- There is no size limit for variables.

## PHP – Variables

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:

- Local variables
- Function parameters
- Global variables
- Static variables

## PHP Local Variables

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

```
<?php
$x = 4;
function assignx () {
    $x = 0;
    print "\$x inside function is $x. ";
}
assignx();
print "\$x outside of function is $x. ";
```

?>

This will produce the following result.

\$x inside function is 0.

\$x outside of function is 4.

## **PHP Function Parameters**

PHP Functions are covered in detail in PHP Function or Unit. In short, a function is a small unit of program which can take some input in the form of parameters and does some processing and may return a value.

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be:

```
<?php
// multiply a value by 10 and return it to the caller
function multiply ($value) {
$value = $value * 10;
return $value;
}
$retval = multiply (10);
Print "Return value is $retval\n";
?>
```

This will produce the following result.

Return value is 100

## **PHP Global Variables**

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished, conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider



an example:

```
<?php
$somevar = 15;
function addit() {
GLOBAL $somevar;
$somevar++;
print "Somevar is $somevar";
}
addit();
?>
```

This will produce the following result.

Somevar is 16

### **PHP Static Variables**

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword `STATIC` in front of the variable name.

```
<?php
function keep_track() {
STATIC $count = 0;
$count++;
print $count;
print " ";
}
keep_track();
keep_track();
keep_track();
```

?>

This will produce the following result.

1

2

3

## OPERATOR

Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

### Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$\$x + \$y$	Sum of $\$x$ and $\$y$
-	Subtraction	$\$x - \$y$	Difference of $\$x$ and $\$y$
*	Multiplication	$\$x * \$y$	Product of $\$x$ and $\$y$
/	Division	$\$x / \$y$	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \$y$	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \$y$	Result of raising $\$x$ to the $\$y$ 'th power (Introduced in PHP 5.6)

### Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a

variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Equivalent	Description
X=Y	X=Y	The left operand gets set to the value of the expression on the right
X+=Y	X=X+Y	Addition
X-=Y	X=X-Y	Subtraction
X*=Y	X=X*Y	Multiplication
X/=Y	X=X/Y	Division
X%=Y	X=X%Y	Modulus Division

### Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	\$x == \$y	Returns true if \$x is equal to \$y
===	Identical	\$x === \$y	Returns true if \$x is equal to \$y, and they are of the same type
!=	Not equal	\$x != \$y	Returns true if \$x is not equal to \$y
<>	Not equal	\$x <> \$y	Returns true if \$x is not equal to \$y
!==	Not identical	\$x !== \$y	Returns true if \$x is not equal to \$y, or they are not of the same type
>	Greater than	\$x > \$y	Returns true if \$x is greater than \$y
<	Less than	\$x < \$y	Returns true if \$x is less than \$y
>=	Greater than or equal to	\$x >= \$y	Returns true if \$x is greater than or equal to \$y
<=	Less than or equal to	\$x <= \$y	Returns true if \$x is less than or equal to \$y

## PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value. The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++\$x	Pre-increment	Increments \$x by one, then returns \$x
\$x++	Post-increment	Returns \$x, then increments \$x by one
--\$x	Pre-decrement	Decrements \$x by one, then returns \$x
\$x--	Post-decrement	Returns \$x, then decrements \$x by one

## Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Results
and	AND	\$x and \$y	True if both \$x and \$y are true
or	OR	\$x or \$y	True if either \$x or \$y is true
xor	XOR	\$x xor \$y	True if either \$x or \$y is true, but not both
&&	AND	\$x && \$y	True if both \$x and \$y are true
	OR	\$x    \$y	True if either \$x or \$y is true
!	NOT	!\$x	True if \$x is not true

## PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Results
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation Assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

## Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Results
+	Union	$\$x + \$y$	Union of $\$x$ and $\$y$
==	Equality	$\$x == \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs
===	Identity	$\$x === \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs in the same order and of the same types
!=	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
<>	Inequality	$\$x <> \$y$	Returns true if $\$x$ is not equal to $\$y$
!==	Non-identity	$\$x !== \$y$	Returns true if $\$x$ is not identical to $\$y$

## Expressions

An expression is a bit of PHP that can be evaluated to produce a value. The simplest expressions are literal values and variables. A literal value evaluates to itself, while a variable evaluates to the value stored in the variable. More complex expressions can be formed using simple expressions and operators.

An operator takes some values (the operands) and does something (for instance, adds them together). Operators are written as punctuation symbols—for instance, the + and - familiar to us from math. Some operators modify their operands, while most do not.

## Flow Control

The if, elseif ...else and switch statements are used to take decision based on the different condition. PHP supports following three decision making statements

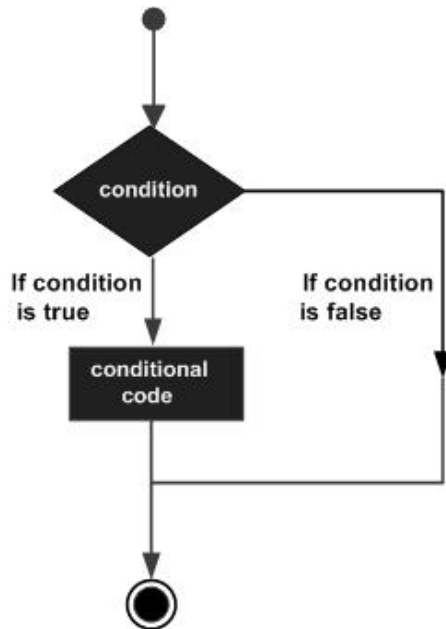


Figure: flowchart of if else

**if...else statement:** use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

### Syntax

```
if (condition)
```

```
code to be executed if condition is true;
```

```
else
```

```
code to be executed if condition is false;
```

**elseif statement:** is used with the if...else statement to execute a set of code if one of the several conditions is true

### Syntax

```
if (condition)
```

```
code to be executed if condition is true;
```

```
elseif (condition)
```

```
code to be executed if condition is true;
```

```
else
```

code to be executed if condition is false;

**Switch statement:** is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

### Syntax

```
switch (expression){  
    case label1:  
code to be executed if expression = label1;  
        break;  
    case label2:  
code to be executed if expression = label2;  
        break;  
    default:  
code to be executed  
        if expression is different  
        from both label1 and label2;  
}
```

### Loops

PHP loops repeat the execution a block of code if specified condition is true. The same block of code to run over and over again in a row, Instead of adding several almost equal code-lines in a script, we can use loops to perform a task.

In PHP, there are following looping statements:

**while - loops** through a block of code as long as the specified condition is true

**do...while - loops** through a block of code once, and then repeats the loop as long as the specified condition is true

**for - loops** through a block of code a specified number of times

**foreach - loops** through a block of code for each element in an array

## PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

### Syntax

```
while (condition is true) {  
    code to be executed;  
}
```

### Example

The example below first sets a variable \$x to 1 (\$x = 1). Then, the while loop will continue to run as long as \$x is less than, or equal to 5 (\$x <= 5). \$x will increase by 1 each time the loop runs (\$x++):

```
<?php  
$x = 1;  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
>
```

This will produce the following result:

```
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5
```

### for Loops

PHP for loops execute a block of code a specified number of times. The for loop is used when you know in advance how many times the script should run.

### Syntax



```
for (init counter; test counter; increment counter) {  
    code to be executed;  
}
```

### **Parameters**

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

### **Example**

The example below displays the numbers from 0 to 10:

```
<?php  
for ($x = 0; $x <= 10; $x++) {  
    echo "The number is: $x <br>";  
}  
?>
```

This will produce the following result:

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 4  
The number is: 5  
The number is: 6  
The number is: 7  
The number is: 8  
The number is: 9  
The number is: 10

## PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

### Syntax

```
foreach ($array as $value) {  
    code to be executed;  
}
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array (\$colors):

### Example

```
<?php  
$colors = array("red", "green", "blue", "yellow");  
foreach ($colors as $value) {  
    echo "$value <br>";  
}  
?>
```

This will produce the following result:

```
red  
green  
blue  
yellow
```

### Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one or more input in the form of parameter and does some processing and returns a value. There are many functions like fopen(), fread(), gettype(), print\_r(), var\_dump etc. as Built-in functions in PHP. It also gives an

option to create userdefine functions as well. There are two parts of a function which are as follow

- Creating a PHP Function
- Calling a PHP Function

In fact, there are more than 1000 of built-in library functions created for different area and it can call according to developer requirement. Hence, function is a self-contained block of code that performs a specific task.

### **Creating a User Defined Function in PHP**

A user-defined function declaration starts with the word function followed by function name shown below:

Syntax

```
function functionName() {  
    code to be executed;  
}
```

A function name can start with a letter or underscore (but not a number). The function name that reflects what the function does will make effective code.

### **Calling a PHP Function**

A user can call php function by its name.

Syntax:

```
function name();
```

### **Example**

```
<?php  
function familyName($fname) {  
    echo "$fnameRefsnes.<br>";  
}  
familyName("Ram");  
familyName("Hari");
```

```
familyName("Shyam");  
familyName("Kamal");  
?>
```

This will produce the following result:

```
Ram  
Hari  
Shyam  
Kamal
```

### **Returning Value from user define function**

The function of the type Arguments with return values will send arguments from the calling function to the called function and expects the result to be returned back from the called function back to the calling function.

To assure a high degree of portability between programs a function should generally be coded without involving any input output operations.

For example, different programs may require different output formats for displaying the results. These shortcomings can be overcome by handing over the result of a function to its calling function where the returned value can be used as required by the program.

### **Example**

```
<?php  
    function addFunction($num1, $num2) {  
        $sum = $num1 + $num2;  
        return $sum;  
    }  
    $return_value = addFunction(10, 20);  
  
    echo "Returned value from the function: $return_value";  
?>
```

This will produce the following result:

Returned value from the function: 30

### **Scope of Variable:**

Variable scope is known as its boundary within which it can be visible or accessed from code. In other words, it is the context within which a variable is defined. There are only two scopes available in PHP namely local and global scopes.

- Local variables (local scope)
- Global variables (special global scope)
- Static variables (local scope)
- Function parameters (local scope)

When a variable is accessed outside its scope it will cause PHP error Undefined Variable.

#### **1. Local Scope Variables**

A local scope is a restricted boundary of a variable within which code block it is declared. That block can be a function, class or any conditional span. The variable within this limited local scope is known as the local variable of that specific code block.

The following code block shows a PHP function. We have declared a variable \$count inside this function. This variable is said to be a local variable of this function and it is in the local scope of the function block.

```
<?php
function calculate_count() {
$count = 5;
//will print 5; the value of local variable
echo $count++;
}
?>
```

Local variables will be destroyed once the end of the code block is reached. Hence the same named variables can be declared within different local scopes.

## 2. Global Scope Variables

As its name, the global scope provides widespread access to the variable declared in this scope. Variables in global scope can be accessed from anywhere from outside a function or class independent of its boundary.

PHP global variables can be defined by using global keyword. Global variables can accessed inside a function by using prefix the global keyword with the variable. The following code shows a code block to learn how to use the global keyword with a PHP variable to declared it as a global variable.

```
<?php
$count = 0;
function calculate_count() {
    global $count;
    // will print 0 and increment global variable
    echo $count++ . "<br/>";
}
calculate_count();
echo $count;
?>
```

PHP has a predefined superglobal variable called \$GLOBALS. It is an associative array with the name of the variable as key and value as the array element. This array variable to add an array of PHP variables in a global scope.

Let us change the above example with the global keyword by using \$GLOBALS superglobal to access the variable in global scope.

```
<?php
$count = 0;
function calculate_count() {
```

```
// will print 0 and increment global variable declared outside function
echo $GLOBALS["count"]++ . "<br/>";
}
calculate_count();
echo $count;
?>
```

### 3. Static Variables (local scope)

A static variable is again a variable with local scope. But the difference with the regular local variable is that it is not destroyed outside the scope boundary. A variable can be defined by using the ‘static’ keyword inside a function.

A static variable does not lose its value when the program execution goes past the scope boundary. But it can be accessed only within that boundary. Let me demonstrate it using the following example code,

```
<?php
function counter()
{
    static $count = 0;
    echo $count;
    $count++;
}
?>
```

The above counter function has the static variable ‘count’ declared within its local scope. When the function execution is complete, the static count variable still retains its value for further computation. Every time the counter function is called, the value of count is incremented. The count value is initialized only once on the first call.

### 4. Function Parameters (Local Scope)

Function parameters (arguments) are local variables defined within the local scope

of the function on which it is used as the argument.

## Scope and File Includes

The boundary of file includes does not demarcate the scope of variables. The scope of a variable is only governed by the function block and not based on the file include. A variable can be declared in a PHP file and used in another file by using 'include' or 'require' that file.

## Function Inside Function or Class

Remember that the scope in PHP is governed by a function block. Any new function declared anywhere starts a new scope. If an anonymous function is defined inside another function, the anonymous function has its own local scope.

```
<?php
function foo() {
    $fruit = 'apple';
    $bar = function () {
        // $fruit cannot be accessed inside here
        $animal = 'lion';
    };

    // $animal cannot be accessed outside here
}
?>
```

In the above example code, \$fruit variable is restricted to the outer function and its scope does not span inside the anonymous inner function. The same way, \$animal which is declared inside is not accessible in the outer function as its scope boundary is restricted to the inner function.

## Use of default parameters in functions

```
<?php
function makecoffee($type = "cappuccino")
```



```

{
    return "Making a cup of $type.\n";
}
echo makecoffee();
echo makecoffee(null);
echo makecoffee("espresso");
?>

```

**This will produce the following result:**

Making a cup of cappuccino.

Making a cup of.

Making a cup of espresso.

### **Passing function parameters by reference**

```

<?php
function add_some_extra(&$string)
{
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str; // outputs 'This is a string, and something extra.'
?>

```

### **Array:**

An array is a special variable, which can hold more than one value at a time. An array can hold many values under a single name, and also can access the values by referring to an index number.

### **Create an Array in PHP**

In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

- Indexed arrays - Arrays with a numeric index
- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

## PHP Indexed Arrays

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota");
```

or the index can be assigned manually:

```
$cars[0] = "Volvo";
```

```
$cars[1] = "BMW";
```

```
$cars[2] = "Toyota";
```

The following example creates an indexed array named \$cars, assigns three elements to it, and then prints a text containing the array values:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

**This will produce the following result:**

I like Volvo, BMW and Toyota.

## Object:

ObjectsPHP is an object-oriented language, although it does not have to be used as one, since most PHP functions are not object oriented. In object-oriented programming, a class is a definition of an object, whereas an object is an instance of an object, meaning that from one class you can create many objects.

```
class Student {
    // constructor
    public function __construct($first_name, $last_name) {
        $this->first_name = $first_name;
```

```

        $this->last_name = $last_name;
    }
    public function say_name() {
        echo "My name is " . $this->first_name . " " . $this->last_name . ".\n";
    }
}
$alex = new Student("Hari", "Badhur");
$alex->say_name();
class MathStudent extends Student {
    function sum_numbers($first_number, $second_number) {
        $sum = $first_number + $second_number;
        echo $this->first_name . " says that " . $first_number . " + " . $second_number
        . " is " . $sum;
    }
}
$eric = new MathStudent("Ram", "Lal");
$eric->say_name();
$eric->sum_numbers(3, 5);

```

**This will produce the following result:**

My name is HariBadhur.

My name is RamLal.

Eric says that 3 + 5 is 8

### **String:**

A string is a sequence of characters, like "Hello world!".The string variables can contain alphanumeric characters. Strings are created when;declare variable and assign string characters to it. It can directly use them with echo statement.String are language construct; it helps capture words.

**The example below returns the length of the string "Hello world!":**

```
<?php
echo strlen("Hello world!");
?>
```

This will produce the following result:

12

**The PHP `str_word_count()` function counts the number of words in a string:**

```
<?php
echo str_word_count("Hello world!");
?>
```

This will produce the following result:

2

## **Date and Time**

PHP date function is an in-built function that simplify working with date data types. The PHP date function is used to format a date or time into a human readable format. It can be used to display the date of article was published, record the last updated a data in a database.

### **PHP Date Syntax**

```
<?php
date(format,[timestamp]);
?>
```

### **Example**

```
<?php
echo date("Y");
?>
```

**Output:** 2019

## Unit: 7

### Working with Form

#### Creating a user Form

```
<form id='register' action='register.php' method='post' accept-charset='UTF-8'>
<fieldset>
<legend>Register</legend>
<input type='hidden' name='submitted' id='submitted' value='1' />
<label for='name' >Your Full Name*: </label>
<input type='text' name='name' id='name' maxlength="50" />
<label for='email' >Email Address*:</label>
<input type='text' name='email' id='email' maxlength="50" />
<label for='username' >UserName*:</label>
<input type='text' name='username' id='username' maxlength="50" />
<label for='password' >Password*:</label>
<input type='password' name='password' id='password' maxlength="50" />
<input type='submit' name='Submit' value='Submit' />
</fieldset>
</form>
```

#### Hidden filed for save state

The hidden value of type defines a form field that is never displayed to the user. The user cannot change the value of the field, or interact with it. When the user submits the form, all of the data they have entered is sent, including the data stored invisibly in the hidden fields.

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

#### Redirecting user

Redirection in PHP can be done using the header() function. To setup a simpler redirect, simply create an index.php file in the directory you wish

to redirect from with the following content:

```
<?php header("Location: http://www.redirect.to.url.com/"); ?>
```

Where 'http://www.redirect.to.url.com/' is the URL you wish the users to be redirected too. This can also be a file, like so:

```
<?php header("Location: anotherDirectory/anotherFile.php"); ?>
```

Files can be of any type including but not limited to HTML, python, php, cgi, perl, and compiled cgi programs.

### **Sending mail on form submission**

The PHP code below is very basic - it will capture the form fields specified in the HTML form above (first\_name, last\_name, email, telephone and comments). The fields are then sent off to your email address in plain text.

#### **Example Form**

First Name *	<input type="text"/>
Last Name *	<input type="text"/>
Email Address *	<input type="text"/>
Telephone Number	<input type="text"/>
Comments *	<input type="text"/>

```
<?php
if(isset($_POST['email'])) {
    // EDIT THE 2 LINES BELOW AS REQUIRED
    $email_to = "you@yourdomain.com";
    $email_subject = "Your email subject line";
```

```

function died($error) {
    // your error code can go here
    echo "We are very sorry, but there were error(s) found with the form you
submitted. ";
    echo "These errors appear below.<br /><br />";
    echo $error."<br /><br />";
    echo "Please go back and fix these errors.<br /><br />";
    die();
}
// validation expected data exists
if(!isset($_POST['first_name']) ||
    !isset($_POST['last_name']) ||
    !isset($_POST['email']) ||
    !isset($_POST['telephone']) ||
    !isset($_POST['comments'])) {
    died('We are sorry, but there appears to be a problem with the form you
submitted.');
```

```

$string_exp = "/^[A-Za-z .'-]+$"/;
if(!preg_match($string_exp,$first_name)) {
    $error_message .= 'The First Name you entered does not appear to be valid.<br
/>';
}
if(!preg_match($string_exp,$last_name)) {
    $error_message .= 'The Last Name you entered does not appear to be valid.<br
/>';
}
if(strlen($comments) < 2) {
    $error_message .= 'The Comments you entered do not appear to be valid.<br />';
}
if(strlen($error_message) > 0) {
    died($error_message);
}
$email_message = "Form details below.\n\n";
function clean_string($string) {
    $bad = array("content-type","bcc:","to:","cc:","href");
    return str_replace($bad,"",$string);
}
$email_message .= "First Name: ".clean_string($first_name)."\n";
$email_message .= "Last Name: ".clean_string($last_name)."\n";
$email_message .= "Email: ".clean_string($email_from)."\n";
$email_message .= "Telephone: ".clean_string($telephone)."\n";
$email_message .= "Comments: ".clean_string($comments)."\n";
// create email headers
$headers = 'From: '.$email_from."\r\n".
'Reply-To: '.$email_from."\r\n" .
'X-Mailer: PHP/' . phpversion();

```



```
@mail($email_to, $email_subject, $email_message, $headers);
```

```
?>
```

```
<!-- include your own success html here -->
```

Thank you for contacting us. We will be in touch with you very soon.

```
<?php
```

```
}
```

```
?>
```

## Working with file uploading

The following HTML code below creates an uploader form. This form is having method attribute set to post.

```
<?php
```

```
if(isset($_FILES['image'])) {
```

```
    $errors = array();
```

```
    $file_name = $_FILES['image']['name'];
```

```
    $file_size = $_FILES['image']['size'];
```

```
    $file_tmp = $_FILES['image']['tmp_name'];
```

```
    $file_type = $_FILES['image']['type'];
```

```
    $file_ext = strtolower(end(explode('.', $_FILES['image']['name'])));
```

```
    $extensions = array("jpeg", "jpg", "png");
```

```
    if(in_array($file_ext, $extensions) === false) {
```

```
        $errors[] = "extension not allowed, please choose a JPEG or PNG file.";
```

```
    }
```

```
        if($file_size > 2097152) {
```

```
            $errors[] = 'File size must be exactly 2 MB';
```

```
        }
```

```
        if(empty($errors) === true) {
```

```
            move_uploaded_file($file_tmp, "images/" . $file_name);
```

```
            echo "Success";
```

```

    }else{
        print_r($errors);
    }
}
?>
<html>
<body>
<form action="" method="POST" enctype="multipart/form-data">
<input type="file" name="image" />
<input type="submit"/>
</form>
</body>
</html>

```

## Working with Session

You might have multiple accounts on numerous websites such as online shopping, social media, professional network, etc. While logging in to these dynamic websites, you need to enter your required details including username, password, location, etc. which creates a session at the server. A PHP session is created with the `session_start()` function and is destroyed with the `session_destroy()` function. A PHP global variable, known as `$_SESSION`, is used to set values to session variables. We can unset all values set to session variables using the `session_unset()` function.

```

<?php
session_start(); //start the PHP_session function
if(isset($_SESSION['page_count']))
{
    $_SESSION['page_count'] += 1;
}
else

```

```
{
    $_SESSION['page_count'] = 1;
}
echo 'You are visitor number ' . $_SESSION['page_count'];
?>
```

## Unit: 8

### Concept of database

#### What is data?

Data is a raw facts collected from environment (surrounding area) about physical phenomena (visible) or business transactions. Data can be in any form, numerical, textual, and graphical, image, sound, video etc.

#### What is information?

Refined or processed data that has been transformed into meaningful and useful form for specific users is called information.

#### Data Hierarchy

A computer system organizes data in hierarchy that begins with bits, and processed to character, fields, records, file and database in figure below.

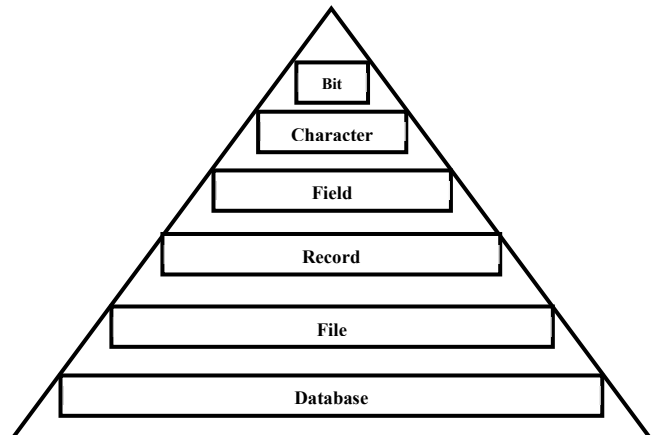


Figure : data hierarchy

#### What is a database?

A database is a repository (storehouse) of data, designed to support efficient data storage, retrieval and maintenance. Binary files, documents, images, videos, relational data, multidimensional data, transactional data, analytical data, or geographical data are stored in a database. For example, University database maintains information about students, courses and grades in university.

## **What is Database System?**

Application software that uses DBMS for data management is called database system. Database system consists of database, Database Management System and application programs.

The Ms Word is just an application program but it is not database system because it does not use DBMS for the purpose of managing data.

## **What is a database management system?**

A database management system is a set of software tools that control, access, organize, store, manage, retrieve and maintain data in a database.

Main goal of Database Management System is to hide underlying complexities of data management from users and provide easy interface to them. Oracle, Sybase, Microsoft SQL Server, Ms-Access etc are the example of DBMS.

## **Components of DBMS**

- **Users:** - User is a person who has to authority to create, design, modifying, updating the database and to access data in the database is called user. Some users have to full authority and some users have to limited authority to controlling, organizing and managing the database. Application Programmers, end users, DBA and system Analyst are four types of users in DBMS. The Application programmers develop the application programs, the end users access the database through application programs, the DBA is responsible for the design, construction, and maintenance of a database and System Analyst determines the requirement of end users
- **Software:** - DBMS, operating system, network software (if necessary) and the application software are necessary for Database system.
- **Hardware:** - Hardware of the system can range from a PC to a network of computers. It also includes various storage devices, like hard disk, and input and output devices.
- **Data:** - Numerical and non numerical data are necessary for the database system. Numerical data are floating and decimal and non numerical data are characters, data or logical (true or false).

## **Application of DBMS**

1. Banking
2. Universities
3. Railway Reservation
4. Airlines
5. Telecommunication
6. Finance
7. Sales

## **Advantage of DBMS**

1. Reduction in data redundancy :
2. Reduction in inconsistency:
3. Sharing data
4. Enforcement of standard:
5. Improvement in data security
6. Maintenance of data integrity
7. Better interaction with users
8. Efficient system

## **Disadvantage**

1. Problems associated with centralization.
2. Cost of software
3. Cost of hardware
4. Complexity of backup and recovery

## **Database Manager**

A database manager (DB manager) is a computer program, or a set of computer programs, that provide basic database management functionalities including creation and maintenance of databases. Database managers have several capabilities including the ability to back up and restore, attach and detach (remove), create, delete and rename the database.

**Interaction with file manager:** - The raw data is stored on the disk using the file

system. The database manager translates the various DML statements into low level file system commands. Thus the database manager is responsible for the actual storing, retrieving and updating of data in the database.

**Integrity enforcement (implement):** -The data values stored in the database must satisfy certain types of consistency constraints. Example, the balance of a bank account may never all below prescribed amount such as Rs 300. These constraints must have specified clearly by the DBA.

**Security enforcement:**-It is not allowed for every user of the database needs to have access to the entire content of the database. It is the job of the database manager to enforce (implement) these security requirements.

**Backup and recovery:** It is the responsibility of the database manager to find defect (fault) the failures system and restore the database. The failure system such as disk crash, power failure and software write errors etc, may lost the data. But, the database manager maintain backup and recovers.

**Concurrency control:** - When several users update the database concurrently the consistency of the data may no longer be preserved. It is necessary for the system to control the interaction (communication) among the concurrent users.

### **Database user**

1. End users
2. Application programmers
3. System Analyst
4. Database Administrator (DBA)

**1. End Users:** End users are the users, who use the applications developed. End users need not know about the working, database design, the access mechanism etc. They just use the system to get their task done. End users are of two types:

- a) Direct users
- b) Indirect users

**a) Direct users:** Direct users are the users who see the computer, database system directly, by following instructions provided in the user interface.

- b) **Indirect users:** Indirect users are those users, who desire benefit from the work of DBMS indirectly.
2. **Application programmers:** These users write application programs to interact (cooperate) with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language.
  3. **System Analyst:** System Analyst determines the requirement of end users, especially naïve (immature) end users and develops specifications for transactions that meet these requirements. System Analyst plays a major role in database design, its properties; the structure prepares the system requirement statement, which involves the feasibility aspect, economic aspect, technical aspect etc. of the system
  4. **Database Administrator (DBA):** Database Administrator (DBA) is the person who makes the strategic and policy decisions regarding the data of the enterprise, and who provide the necessary technical support for implementing these decisions. The person having the control of data and programs accessing over the system is called the database administrator (DBA). DBA is responsible for defining procedures to recovery the database from failures due to human, natural or hardware causes with minimal loss of data. Therefore, DBA is responsible for overall control of the system at a technical level. The functions of the DBA are as follows:

**Schema definition:** - The DBA responsible for the creation of the original database schema (plan). He is also responsible for the definition and implementation of the internal level.

**Storage structure and access method definition:-** The function of the DBA is the creation of the storage structure and access method to be used for the optimum performance of the DBMs. This is accomplished by writing a set of definition which is translated by the data storage definition language compiler.

**Granting of authorization for data access:-** DBA next function is the granting permission of different types of authorization for data access to the various users of the database. The DBA is responsible for granting permission to the users of the



database and stores the profile of each user in the database.

**Integrity constraint specification:** - DBA's next function is to keep the constraints in a special system structure that is consulted by the Database Manager.

### **Needs of Database**

A database is needed because it manages data efficiently and allows users to perform multiple tasks with ease. A logical grouping of related files are called database. Use of this system increases efficiency of business operations and reduces overall costs. There are two types of approach of data management,

- File management system
- Database management system

Organizations have different data which need to be processed for getting information quickly. Information is vital resources in development activities of any society. Information is obtained through processing of data. Information is so important for the decision making. Getting the information quickly from the traditional (manual) system is difficult. Database makes the information retrieval quickly. Database provides the information according to the requirement of user with the single click.

### **Organization of database**

A database management system is important because it manages data efficiently and allows users to perform multiple tasks with ease. A database management system stores, organizes and manages a large amount of information within a single software application. Some of the data that are easily managed with this type of system include: employee records, student information, payroll, accounting, project management, inventory and library books. There are four main types of computerized database organization:

- Flat file
- Hierarchical
- Relational
- Network

## Flat database organization

A “flat file” is a plain text or mixed text and binary file which usually contain one record per line or 'physical' record. Such as a list of names, addresses, and phone numbers written on a sheet of paper is a flat file database. This can also be done with any typewriter or word processor. A spreadsheet or text editor program may be used to implement flat file databases.

### Flat File Structure

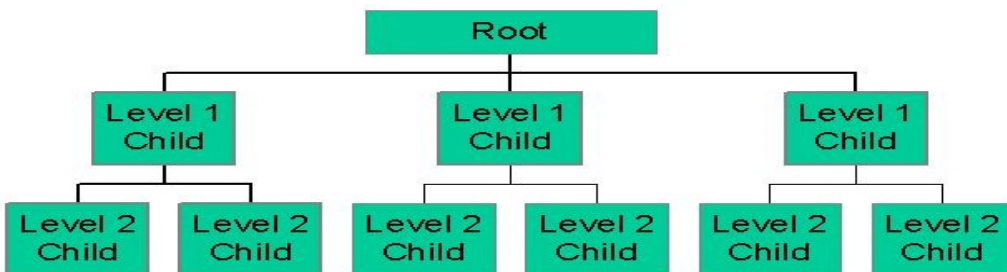
Tax No.	Street Address	Subdiv	Block/Lot	Owner 1	Owner 2	Improved	GC Warbler
234	10 Lone Oak	RobRoy	A/116	Verdi, G.	Rossini, G.	Yes	No
235	12 Lone Oak	RobRoy	A/118	Wagner, R.	Weber, C.	No	Yes
236	101 Madrone	LiveOak	B/14	Hendrix, J.	Morrison, J.	Yes	Yes

All individual property data is in one file. The Tax No. is required to search the database.

**Figure: flat database organization**

**Hierarchical database organization:** A hierarchical database is a design that uses a one-to-many relationship for data elements. A database organization method that is structured in a hierarchy. In a hierarchical database, records contain groups of parent/child relationships, similar to a tree structure. Hierarchical databases are fast and simple but inflexible (rigid) as the relationship is restricted to one-to-many, only allowing for one parent segment per child.

## Hierarchical database model



**Figure: Hierarchical database model**

## Relational Database organization

A relational database is a collection of data items organized as a set of formally described tables from which data can be accessed easily. Relational databases are one of the most effective type of database organization. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. A relational database is created using the relational model. The DBMS software used in a relational database is called a relational database management system (RDBMS).

### Relational Model

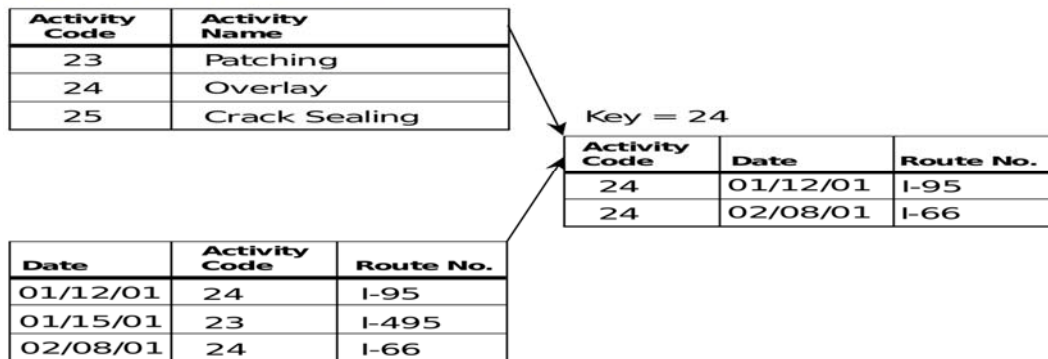


Figure: Relational Model

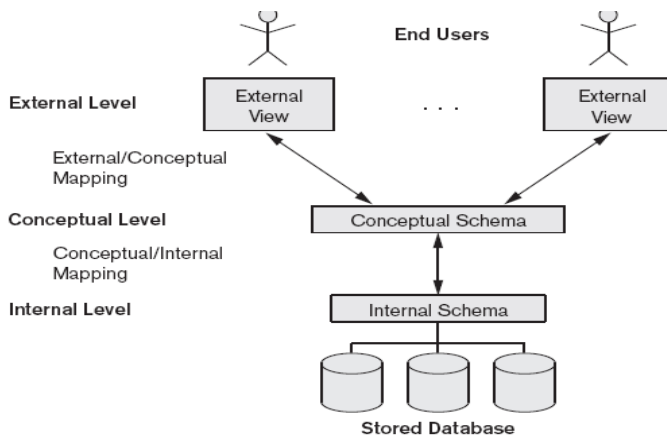


Figure:database architecture

## Network database organization

Network database organization is collection of data item organized in network

basis. It is the extension of the hierarchical database structure. It is also record based representational or implementation database organization.

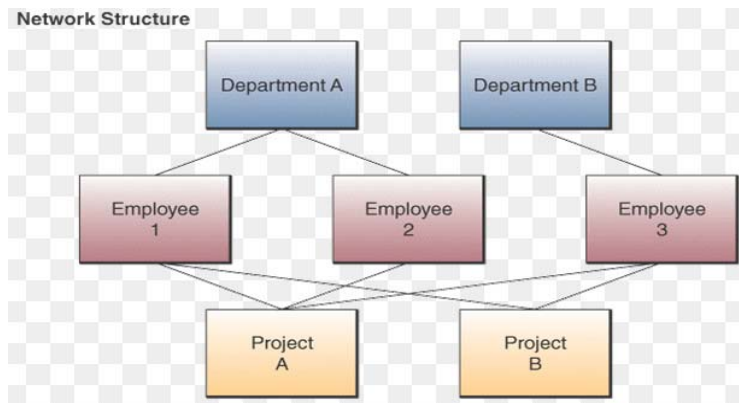


Figure: Network database

### Client server Architecture

Client server is a software architecture in which two processes interact as superior and subordinate. The client process always initiates requests and the server process always responds to requests. Theoretically, client and server can reside on the same machine. Typically, however they run on separate machines linked by some form of communication network usually a local area network.

A client server database system generally refers to a local area network of personal computer. At least one of these PCs is dedicated to server, the database needs of the others which act in a client capacity. The database held on the server. The user interface and application development tools are held on the client machine.

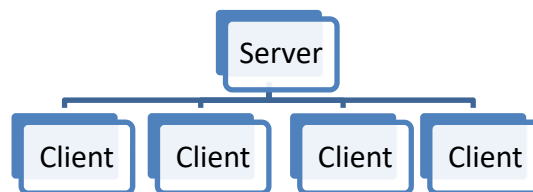


Figure: Client Server Architecture

**Data dictionary:** - A data dictionary would have provided the definition of data items, how they fit into the data structure and relate to other entities in the database. Data dictionary provides documentations which are valuable to end users, managers

and programmer. A data dictionary can be a great asset the DBA for database design, implementation, maintenance and project planning.

## SQL Language

SQL is a programming language for Relational Databases. SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database. SQL defines the method used to create and manipulate relational database all major platforms using three types of language.

**Data Manipulation Language (DML)** : DML is a Language that enables user to access or manipulate data as organized by appropriate data model. DML consists of SQL statement for operating on the data (inserting, modifying, deleting and retrieving data) in table which already exist.

**DDL(Data Definition Language)** : is a Language, usually run of a database management system that is used to define all attributes and properties of a database, especially row layouts, columns definition, key columns, file location and storage strategy. DDL consists of the schema (creating, modifying and Dropping tables, indexes etc)

**DCL (Data Control Language)** : consists Data recovery and Concurrency, Data security and data integrity constraints.Examples of DCL commands include:

- GRANT to allow specified users to perform specified tasks.
- REVOKE it is remove the user accessibility to database object.

The operations for which privileges may be granted to or revoked from a user or role apply to both the Data definition language (DDL) and the Data manipulation language (DML), and may include CONNECT, SELECT, INSERT, UPDATE, DELETE, EXECUTE, and USAGE.

**Some syntax of SQL (CRUD syntax) means:**

Create = INSERT,

Read = SELECT,

Update = UPDATE,

Delete = DELETE

**Create: - Create statement creates the database and the Table in the DBMS.**

Creating database:-

syntax:-

```
create database database name
```

Example:-

```
create database school
```

**To use created database**

Syntax:-

```
use database database-name
```

Example:-

```
use database school
```

To create table in database

Syntax

```
CREATE TABLE table name
```

```
(  
column_name1 data_type(size),  
column_name2 data_type(size),  
column_name3 data_type(size),  
....  
);
```

**Example**

```
create table student
```

```
(  
rollno int,  
name varchar(255),
```

```
address varchar(255),  
fathersname varchar(255)  
);
```

### **INSERT INTO statement**

The INSERT INTO statement is used to insert new records in a table.

Syntax

```
Insert into table name(column 1, column2.....column n)
```

```
Values('values1', 'values2'.....'values n');
```

Example

```
insert into student(Name, Rollno, fathename, qualification)  
values ('kumar', '23', 'keshab', 'computer engineering running');
```

### **Alter statement**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
```

```
ADD column_name datatype
```

Example

```
ALTER TABLE student
```

```
ADD DateOfBirth date
```

### **The SQL UPDATE Statement**

The UPDATE statement is used to update records in a table.

The UPDATE statement is used to update existing records in a table.

### **SQL UPDATE Syntax**

```
UPDATE table_name
```

SET column1=value1,column2=value2,...

WHERE some\_column=some\_value;

### **Example**

update student

set Name='uma',fathername='bhawani'

where Name='prajwal';

### **The SQL DELETE Statement**

The DELETE statement is used to delete rows in a table.

SQL DELETE Syntax

DELETE FROM *table\_name*

WHERE *some\_column=some\_value*;

Example

delete from student

where Rollno='5' and Name='pp';

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE *table\_name*

DROP COLUMN *column\_name*

### **Example**

ALTER TABLE student

DROP COLUMN Date Of Birth

### **Database security and Back up**

#### **Database security**

Data security is the practice of keeping data protected from corruption and unauthorized access. Database security allows or not allows user actions on the database and objects within it. It protects a database from unintended activity. Unintended activity can be categorized as authenticated misuse, malicious (harmful) attacks or inadvertent (unintentional) mistakes made by authorized



individuals.

Three main securities objective are: secrecy, integrity, availability.

- **Secrecy:** Users should not be able to see things they are not supposed to (imaginary). E.g., a student can't see other students' grades.
- **Integrity:** Users should not be able to modify things they are not supposed to. E.g., only instructors can assign grades.
- **Availability:** Users should be able to see and modify things they are allowed to.

Database security can be defined as a system or process by which the "Confidentiality, Integrity, and Availability," or CIA, of the database can be protected. Unauthorized entry or access to a database server signifies a loss of confidentiality; unauthorized alteration to the available data signifies loss of integrity; and lack of access to database services signifies loss of availability. Loss of one or more of these basic factors will have a significant impact on the security of the database.

### **Classification of database security**

- **Physical Security:** Physical security refers to the security of the hardware associated with the system and the protection of the site where the computer reside.
- **Logical security:** Logical security consists of software safeguards for an organization's systems, including user identification and password access authenticating, access rights and authority levels.

**Example of firewall :** firewall, Cisco, other device

### **Encryption**

Encryption is process of transforming readable data into unreadable data. Encryption is the most effective way to achieve data security. A DBMS can use encryption to protect information in certain situations where the normal security mechanisms of the DBMS are not adequate (sufficient). To read an encrypted file, you must have access to a secret key or password that enables you to decrypt it. For example, an intruder may steal storage containing some data or data communication

line. By storing and transmitting data in an encrypted form, the DBMS ensures that such stolen data is not intelligible (easily understood) to the intruder (a person or thing that disturb other). Thus, encryption is a technique to provide privacy of data. In encryption, the message to be encrypted is known as plaintext. The plaintext is transformed by a function that is parameterized by a key. The output of the encryption process is known as the cipher text. Cipher text is then transmitted over the network. In other words, the process of converting the plaintext to cipher text is called as Encryption and process of converting the cipher text to plaintext is called as Decryption. Encryption is performed at the transmitting end and decryption is performed at the receiving end.

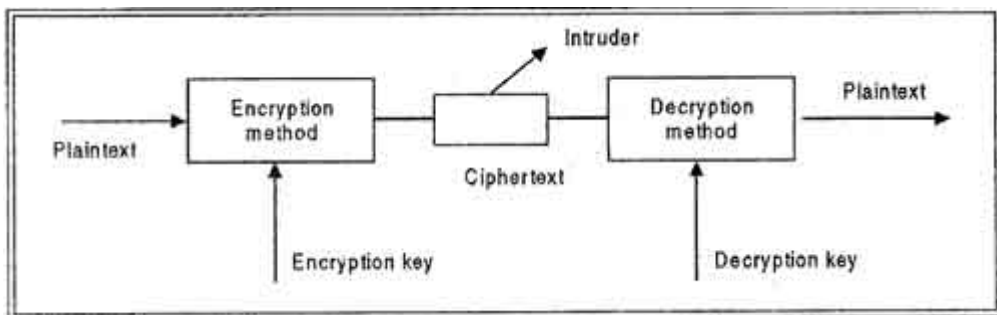


Figure: process of encryption and decryption

## Database backup

Database backup is a way to protect and restore a database. It is performed through database replication and can be done for a database or a database server. Typically, database backup is performed by the RDBMS or similar database management software.

Database administrators can use the database backup copy to restore the database to its operational state along with its data and logs. The database backup can be stored locally or on a backup server.

Database backup is also created/performed to ensure a company's compliance with business and government regulations and to maintain and ensure access to critical/essential business data in case of a disaster or technical outage.

## Unit: 9

# MySQL and PHP

### Introduction

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation. The data in a MySQL database are stored in a table which consists of columns and rows.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994. PHP is a recursive acronym for "PHP: Hypertext Preprocessor". PHP is a server side scripting language that is embedded in HTML.

### Connecting to MySQL with PHP

MySQL, a highly popular DBMS that can power projects of all sizes. MySQL's ability to handle huge volumes of data without breaking a sweat is one of its biggest selling points. how to connect MySQL Database using PDO.

### How to connect to MySQL using PHP

We can refer several methods for connecting to a MySQL database using PHP:

- MySQL Improved (*mysqli*) extension
- PDO (PHP Data Objects)
- Legacy MySQL (*mysql\_*) functions
- Connecting to a remote MySQL database using PHP

### CONNECTING TO MYSQL USING THE MYSQL IMPROVED EXTENSION

The MySQL Improved extension uses the *mysqli* class, which replaces the set of legacy MySQL functions.

To connect to MySQL using the MySQL Improved extension, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
<?php
```

```
    $mysqli = new mysqli("localhost", "username", "password", "dbname");
```

```
?>
```

After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations. For example, the following PHP code runs a SQL query that extracts the last names from the *employees* table, and stores the result in the *\$result* variable:

```
<?php
```

```
    $result = $mysqli->query("SELECT lastname FROM employees");
```

```
?>
```

## CONNECTING TO MYSQL USING PDO (PHP DATA OBJECTS)

The MySQL Improved extension can only be used with MySQL databases. PDO, on the other hand, abstracts database access and enables you to create code that can handle different types of databases.

To connect to MySQL using PDO, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
<?php
```

```
    $myPDO = new PDO('mysql:host=localhost;dbname=dbname',  
                    'username', 'password');
```

```
?>
```

2. After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations. For example, the following PHP code runs a SQL query that extracts the last names from the *employees* table, and stores the result in the *\$result* variable:

```
<?php
```

```
    $result = $myPDO->query("SELECT lastname FROM employees");
```

?>

## CONNECTING TO MYSQL USING THE LEGACY MYSQL FUNCTIONS

The original PHP MySQL functions (whose names begin with *mysql\_*) are deprecated in PHP 5.5, and will eventually be removed from PHP. Therefore, you should only use these functions when absolutely necessary for backward compatibility. If possible, use the MySQL Improved extension or PDO instead.

To connect to MySQL using the legacy PHP MySQL functions, follow these steps:

1. Use the following PHP code to connect to MySQL and select a database. Replace *username* with your username, *password* with your password, and *dbname* with the database name:

```
<?php
```

```
mysql_connect('localhost','username','password');  
mysql_select_db("dbname");
```

```
?>
```

2. After the code connects to MySQL and selects the database, you can run SQL queries and perform other operations. For example, the following PHP code runs a SQL query that extracts the last names from the *employees* table, and stores the result in the *\$result* variable:

```
<?php
```

```
    $result = mysql_query('SELECT lastname FROM employees');
```

```
?>
```

## CONNECTING TO A REMOTE MYSQL DATABASE USING PHP

The previous examples all assume that the PHP script runs on the same server where the MySQL database is located. But what if you want to use PHP to connect to a MySQL database from a remote location? For example, you may want to connect to your A2 Hosting database from a home computer or from another web server.

**To do this, you need to do two things:**

- On the A2 Hosting server, enable the connecting IP address for remote access.

For information about how to do this, please see [this article](#).

If you do not add your IP address to the list of permitted remote access hosts, you receive **Access denied** messages when you try to access a MySQL database remotely.

- In your PHP code, change the MySQL connection string to use the A2 Hosting server name instead of *localhost*. For example, the following PHP code uses *mysqli* to connect to the A2 Hosting server *a2ss25.a2hosting.com*:

```
<?php
```

```
    $mysqli = new mysqli("a2ss25.a2hosting.com", "username", "password",  
    "dbname");
```

```
?>
```

## Working with MySQL Data

Simple PHP & MySQL code to add, edit, delete and view data. This kind of system is also referred to **CRUD (Create, Read, Update, and Delete)**.

First of all, we will create a new MySQL database

```
create database test;
```

Then, we will create a new table in database ‘test’. Let us name the table as ‘**users**’.

```
use test;
```

```
CREATE TABLE users (  
id int(11) NOT NULL auto_increment,  
name varchar(100) NOT NULL,  
age int(3) NOT NULL,  
email varchar(100) NOT NULL,  
PRIMARY KEY (id)  
);
```

Now, we will create a **config.php** file which contains database connection code. This code connects to the MySQL database. This file is included in all PHP pages where database connection is necessary.

In below code, the database host name is localhost where username=root and password=root. The database test has been selected.

```
<?php
/*
// mysql_connect("database-host", "username", "password")
$conn = mysql_connect("localhost","root","root")
        or die("cannot connected");
// mysql_select_db("database-name", "connection-link-identifier")
@mysql_select_db("test",$conn);
*/
/**
 * mysql_connect is deprecated
 * using mysqli_connect instead
 */
$databaseHost = 'localhost';
$databaseName = 'test';
$databaseUsername = 'root';
$databasePassword = 'root';
$mysqli = mysqli_connect($databaseHost, $databaseUsername,
$databasePassword, $databaseName);
?>
```

To add data into database, we need an html form.

### **add.html**

```
<html>
<head>
    <title>Add Data</title>
</head>
```

```

<body>
  <a href="index.php">Home</a>
  <br/><br/>

  <form action="add.php" method="post" name="form1">
    <table width="25%" border="0">
      <tr>
        <td>Name</td>
        <td><input type="text" name="name"></td>
      </tr>
      <tr>
        <td>Age</td>
        <td><input type="text" name="age"></td>
      </tr>
      <tr>
        <td>Email</td>
        <td><input type="text" name="email"></td>
      </tr>
      <tr>
        <td></td>
        <td><input type="submit" name="Submit" value="Add"></td>
      </tr>
    </table>
  </form>
</body>
</html>

```

Form action on add.html is add.php. It means that the submitted form data will go to add.php. In add.php, we do a simple validation of checking if the entered name, email & age are empty or not. If they are all filled then the data will be inserted into



database table.

### **add.php**

```
<?php
//including the database connection file
include_once("config.php");
//fetching data in descending order (lastest entry first)
//$result = mysql_query("SELECT * FROM users ORDER BY id DESC"); //
mysql_query is deprecated
$result = mysqli_query($mysqli, "SELECT * FROM users ORDER BY id DESC");
// using mysqli_query instead
?>

<html>
<head>
    <title>Homepage</title>
</head>

<body>
    <a href="add.html">Add New Data</a><br/><br/>

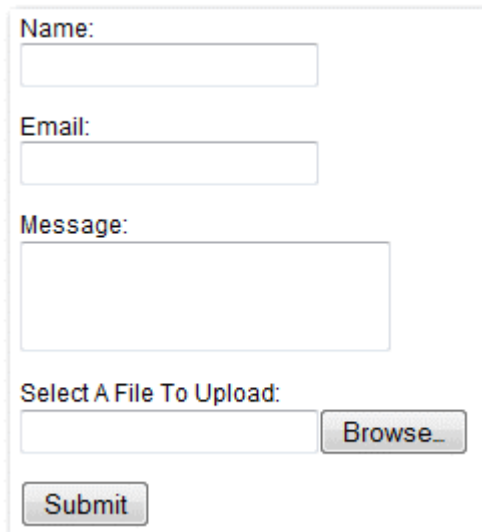
    <table width='80%' border=0>
        <tr bgcolor=#CCCCCC'>
            <td>Name</td>
            <td>Age</td>
            <td>Email</td>
            <td>Update</td>
        </tr>
        <?php
            //while($res = mysql_fetch_array($result)) { // mysql_fetch_array is
deprectated, we need to use mysqli_fetch_array
```

```

while($res = mysqli_fetch_array($result)) {
    echo "<tr>";
    echo "<td>".$res['name']. "</td>";
    echo "<td>".$res['age']. "</td>";
    echo "<td>".$res['email']. "</td>";
    echo "    "<td><a href=\ "edit.php?id=$res[id]">Edit</a> | <a
href=\ "delete.php?id=$res[id]" onClick=\ "return confirm('Are you sure you want
to delete?')\">Delete</a></td>";
    }
?>
</table>
</body>
</html>

```

## File Uploading & Emails



The image shows a web form with the following elements:

- Name:** A text input field.
- Email:** A text input field.
- Message:** A larger text area for entering a message.
- Select A File To Upload:** A text input field with a "Browse..." button to the right.
- Submit:** A button at the bottom of the form.

Figure:file uploading

The following code handles the form submission and email sending functionality using PHP.

- Get the submitted form data using **\$\_POST in PHP**.
- Validate form data to check whether the mandatory fields are not empty.
- Validate email address using **FILTER\_VALIDATE\_EMAIL in PHP**.
- Check the file extension to allow certain file formats (PDF, Image, and MS Word file).
- Upload file to the server.
- Add form data and uploaded file to the email content.
- Send an email with attachment to the specified recipient using **PHP mail()** function.

```
<?php
```

```
$postData = $uploadedFile = $statusMsg = ";
```

```
$msgClass = 'errordiv';
```

```
if(isset($_POST['submit'])){
```

```
    // Get the submitted form data
```

```
    $postData = $_POST;
```

```
    $email = $_POST['email'];
```

```
    $name = $_POST['name'];
```

```
    $subject = $_POST['subject'];
```

```
    $message = $_POST['message']; // Check whether submitted data is not empty
```

```
    y
```

```
    if(!empty($email) && !empty($name) && !empty($subject) && !empty($message)){ // Validate email
```

```
        if(filter_var($email, FILTER_VALIDATE_EMAIL) === false){
```

```
            $statusMsg = 'Please enter your valid email.';
```

```
        }else{
```

```
            $uploadStatus = 1; // Upload attachment file
```

```
            if(!empty($_FILES["attachment"]["name"])){ // File path config
```

```
                $targetDir = "uploads/";
```

```
                $fileName = basename($_FILES["attachment"]["name"]);
```

```
                $targetFilePath = $targetDir . $fileName;
```

```
                $fileType = pathinfo($targetFilePath,PATHINFO_EXTENSION);
```

```

// Allow certain file formats
$allowTypes = array('pdf', 'doc', 'docx', 'jpg', 'png', 'jpeg');
if(in_array($fileType, $allowTypes)){
    // Upload file to the server
    if(move_uploaded_file($_FILES["attachment"]["tmp_name"], $target
FilePath)){
        $uploadedFile = $targetFilePath;
    }else{
        $uploadStatus = 0;
        $statusMsg = "Sorry, there was an error uploading your file.";
    }
}else{
    $uploadStatus = 0;
    $statusMsg = 'Sorry, only PDF, DOC, JPG, JPEG, & PNG files are allow
ed to upload.';
}
} if($uploadStatus == 1){

```

```

// Recipient
$toEmail = 'admin@example.com'; // Sender
$from = 'sender@example.com';
$fromName = 'CodexWorld';
// Subject
$emailSubject = 'Contact Request Submitted by '.$name;
// Message
$htmlContent = '<h2>Contact Request Submitted</h2>
<p><b>Name:</b> '.$name.'</p>
<p><b>Email:</b> '.$email.'</p>
<p><b>Subject:</b> '.$subject.'</p>
<p><b>Message:</b><br/>'.$message.'</p>';
// Header for sender info
$headers = "From: $fromName"." <".$from.">";

```

```

if(!empty($uploadedFile) && file_exists($uploadedFile)){
    // Boundary
    $semi_rand = md5(time());
    $mime_boundary = "==Multipart_Boundary_x{$semi_rand}x";
        // Headers for attachment
    $headers .= "\nMIME-Version: 1.0\n" . "Content-
Type: multipart/mixed;\n" . " boundary=\"{$mime_boundary}\"";
    // Multipart boundary
    $message = "--{$mime_boundary}\n" . "Content-
Type: text/html; charset=\"UTF-8\"\n" .
    "Content-Transfer-Encoding: 7bit\n\n" . $htmlContent . "\n\n";
    // Preparing attachment
    if(is_file($uploadedFile)){
        $message .= "--{$mime_boundary}\n";
        $fp = @fopen($uploadedFile,"rb");
        $data = @fread($fp,filesize($uploadedFile));
        @fclose($fp);
        $data = chunk_split(base64_encode($data));
        $message .= "Content-Type: application/octet-
stream; name=\"".basename($uploadedFile)."\n" .
        "Content-Description: ".basename($uploadedFile)."\n" .
        "Content-
Disposition: attachment;\n" . " filename=\"".basename($uploadedFile).\""; size="
filesize($uploadedFile).";\n" .
        "Content-Transfer-Encoding: base64\n\n" . $data . "\n\n";
    }
    $message .= "--{$mime_boundary}--";
    $returnpath = "-f" . $email;
    // Send email
    $mail = mail($toEmail, $emailSubject, $message, $headers, $returnp
ath);

```



Country

Subject

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}
* {box-sizing: border-box;}
input[type=text], select, textarea {
width: 100%;
padding: 12px;
border: 1px solid #ccc;
border-radius: 4px;
box-sizing: border-box;
margin-top: 6px;
margin-bottom: 16px;
resize: vertical;
}
input[type=submit] {
background-color: #4CAF50;
color: white;
```

```

padding: 12px 20px;
border: none;
border-radius: 4px;
cursor: pointer;
}
input[type=submit]:hover {
background-color: #45a049;
}
.container {
border-radius: 5px;
background-color: #f2f2f2;
padding: 20px;
}
</style>
</head>
<body>
<h3>Contact Form</h3>
<div class="container">
<form action="/action_page.php">
<label for="fname">First Name</label>
<input type="text" id="fname" name="firstname" placeholder="Your name..">
<label for="lname">Last Name</label>
<input type="text" id="lname" name="lastname" placeholder="Your last name..">
<label for="country">Country</label>
<select id="country" name="country">
<option value="australia">Australia</option>
<option value="canada">Canada</option>
<option value="usa">USA</option>
</select>

```



```
<label for="subject">Subject</label>
<textarea id="subject" name="subject" placeholder="Write something.."
style="height:200px"></textarea>
<input type="submit" value="Submit">
</form>
</div>
</body>
</html>
```

## PHP and AJAX

PHP is a script language and interpreter that is freely available and used primarily on LinuxWeb servers. PHP, originally derived from Personal Home Page Tools, now stands for PHP: Hypertext Preprocessor. PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. AJAX (Asynchronous JavaScript and XML )is a technique for creating fast and dynamic web pages.AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

## Image CAPTCHA

CAPTCHA stands for “Completely Automated Public Turing test to tell Computers and Humans Apart”, it’s very common functionality to use at the time of submit data to **prevent machines** access of your website.



Figure: CAPTCHA

## References & Bibliography

Felke-Morris, T. (2013). Web development and design foundations with HTML5

(6th ed). New Delhi : Pearson.

MacCaw, A. (2011). Java Script web applications: Alex MacCaw. O'Reilly.

Meloni Julie C(2010).,Teach Yourself PHP, MySQL and Apache, New Delhi,

Pearson Education Pvt. Ltd. Delhi Holzner, Steven(2008), PHP : the complete reference, New Delhi ,Tata McGraw-Hill

Date, C.J., An Introduction to Database, 7th Edition, Addison Wesley. 2010

Raghu Ramakrishnan, “Database Management System”, Tata McGraw-Hill Publishing Company, 2003.

<http://www.dbta.com/Editorial/Think-About-It/5-Key-Steps-to-Ensuring-Database-Security-95307.aspx>

<https://www.studytonight.com/dbms/architecture-of-database>

[https://www.tutorialspoint.com/dbms/dbms\\_data\\_models.htm](https://www.tutorialspoint.com/dbms/dbms_data_models.htm)

<https://www.wikipedia.com>

<https://www.tutorialspoint.com/dbms/>

<https://www.tutorialspoint.com/php/>