



(Affiliated to Purbanchal University)

Khwopa Engineering College

Libali-2, Bhaktapur



A Complete Manual Of
Free Open Source Programming
BE Computer (Fourth Semester) New Syllabus
2012

Prepared By
Er. Ganesh Ram Suwal
Lecturer
Khwopa Engineering College

Free and Open Source Programming
BEG *CO**

Year: II

Semester: II

Teaching Schedule Hours/Week			Examination Scheme				
Theory	Tutorial	Practical	Internal Assessment		Final		Total
3	1	3	Theory Marks	Practical Marks*	Theory Marks **	Practical Marks	150
			20	50	80	-	

* Continuous

** Duration: 3 hours

Course Objective: To provide basic concept of ‘Free and Open Source Programming’ and its applications.

1. Free and Open Source Software (FOSS) an Overview - (5hrs)

- 1.1 Introduction
- 1.2 The FOSS Philosophy
- 1.3 History and evolution of FOSS
- 1.4 Design Logic, Source Code, Binary Code
- 1.5 Examples of Open Source software products
- 1.6 Emerging applications of FOSS philosophy in various sectors.

2. Classification of Free and Open Source Software - (5hrs)

- 2.1 Free Software
- 2.1 Open Source Software

- 2.3 Proprietary Software
- 2.4 Other existing Software models
- 2.5 Open Standards
- 2.6 Open Content
- 2.7 Benefits and Shortcoming of FOSS
- 2.8 Strengths and weakness of FOSS
- 2.9 Comparison of FOSS and Proprietary software

3. Licensing (4hrs)

- 3.1 Types of licensing
- 3.2 Commercial License versus Open Source License
- 3.3 Open Source Software Licensing
 - Types of OSS licenses
 - OSS licensing strategies

4. Web Basics(3 Hrs)

- 4.1 Web Browsers
- 4.2 Web Servers
- 4.3 Types of Web Pages & its Processing in WWW
- 4.4 HTTP, HTTPS
- 4.5 HTTP Transaction
- 4.6 FTP & its Types.

5. Web Development with HTML & DHTML (6 Hrs)

- 5.1. Introduction to HTML
- 5.2 HTML Assistants, Editors, Convertors, Images and Multimedia, Linking Documents, Tables, Frames, Image Maps, Forms, CSS

6. Introduction to JavaScript (4 Hrs)

- 6.1 Basic Introduction
- 6.2 Functions
- 6.3 Error Handling
- 6.4 Dialog Box
- 6.5 Form Validation

7. Open Source Programming with PHP (10 Hrs)

Introduction

- a. Syntax
- b. Operators
- c. Variables
- d. Constants
- e. Control Structures
- f. Language Constructs and Functions

Arrays

- a. Enumerated Arrays
- b. Associative Arrays
- c. Array Iteration
- d. Multi-Dimensional Arrays
- e. Array Functions

Functions

- a. Syntax
- b. Arguments
- c. Variables
- d. References
- e. Returns
- f. Variable Scope

File Handling

- a. Files
- b. Reading
- c. Writing
- d. File System Functions

8. Databases Connectivity in PHP (4Hrs)

- SQL
- Basic SQL Queries(CRUD)
- Database Connectivity

9. Session and Cookies (4 Hrs)

- Introduction to session

- Create session
- Destroy session
- Cookies

Laboratory

There shall be lab exercises to cover all the theoretical concept of the Free & Open Source Programming.

References

1. “Free and Open Source Software A general Introduction” by Kenneth Wong and Phet Sayo, Published by IOSN APDIP.
2. The Cathedral and the Bazaar; Musings on Linux and Open Source by an Accidental Revolutionary by Eric S. Raymond.
3. HTML, DHTML, JavaScript & PHP, Ivan Bayross (New Edition.)
4. Beginning of PHP, WROX , PHI Publishing House
4. Professional PHP Programming, Jesus M. Castagnetto, Harish Rawat, Deepak T. Veliath

Evaluation Schemes

The questions should cover all the chapters of the syllabus. The marks distribution will be as indicated in the table below.

Chapter	Hour	Marks Distribution
1	5	7
2	5	10
3	4	6
4	3	4
5	6	10
6	4	10
7	10	20
8	4	7
9	4	6
Total	45	80

Note: There may be minor deviation in marks distribution.

CHAPTER 1: INTRODUCTION - Free Open Source Software

F stands for **Free**. Free software comes without a cost to the user. **OSS** stands for Open Source Software.

F.O.S.S. allows an individual the freedom and right to use software licensed under an open source agreement. The individual may use the software as is or may make changes as needed. This use may or may not come with a cost depending on the developer's software philosophy. Open source software can be used, studied and/or modified dependent upon the needs of the user and any modifications are still held to the open source guidelines and criteria.

Proprietary software is on the other end of the spectrum and is another name for non-free software and the coding is closed.

Open Source software is distributed with its source code. The Open Source Definition has three essential features:

- It allows **free re-distribution** of the software without royalties or licensing fees to the author

- It requires that **source code** be distributed with the software or otherwise made available for no more than the cost of distribution
- It allows anyone to **modify** the software or derive other software from it, and to redistribute the modified software under the same terms.

Example of Open Source Software

- Operating Systems
 - ✓ Linux
 - ✓ FreeBSD, OpenBSD, and NetBSD: The BSDs are all based on the Berkeley Systems Distribution of Unix, developed at the University of California, Berkeley. Another BSD based open source project is Darwin, which is the base of Apple's Mac OS X.
- Internet
 - ✓ Apache, which runs over 50% of the world's web servers.
 - ✓ BIND the software that provides the DNS (domain name service) for the entire Internet.
 - ✓ Send mail, the most important and widely used email transport software on the Internet.
 - ✓ Mozilla, the open source redesign of the Netscape Browser
 - ✓ OpenSSL is the standard for secure communication (strong encryption) over the Internet.categories.
- Programming Tools
 - ✓ Zope, and PHP, are popular engines behind the "live content" on the World Wide Web.
 - ✓ Languages:
- Perl
- Python
- Ruby
- Tcl/Tk
 - ✓ GNU compilers and tools
- GCC
- Make
- Autoconf
- Automake
- etc.

History and Evolution of Free Open Source Software

- **1970s:** UNIX operating system developed at Bell Labs and by a diverse group of contributors outside of Bell Labs; later AT&T enforces intellectual property rights and "closes" the code
- **1983:** Richard Stallman founds the Free Software Foundation
- **1993:** Linus Torvalds releases first version of Linux built
- **1997:** Debian Free Software Guidelines released
- **1998:** Netscape releases Navigator in source

Open source software sites

- Free Software Foundation **www.fsf.org**

- Open Source Initiative **www.opensource.org**
- Freshmeat.net
- SourceForge.net
- OSDir.com
- developer.BerliOS.de
- Bioinformatics.org
- See also individual project sites; e.g., **www.apache.org**; **www.cpan.org**; etc.

Copyright: The exclusive right to produce or reproduce (copy), to perform in public or to publish an original literary or artistic work. Many countries have expanded the definition of a "literary work" to include computer programs or other electronically stored information.

License: A special permission to do something on, or with, somebody else's property which, were it not for the license, could be legally prevented or give rise to legal action in tort or trespass.

(From DUHAIME'S LAW DICTIONARY)

Free, Open Source and Copyright are NOT legally defined

The FOSS Philosophy is based primarily on the need for software to be free and unencumbered: there must be "open distribution and open modification"

CHAPTER 2: Free vs Open Source

➤ **Free Software Foundation (FSF)**

- Non-profit organization, founder: Richard M. Stallman, founded in 1985
- Free as in free speech, not as in free beer
- Principal organizational sponsor of the GNU Project
- **www.fsf.org**

➤ **Open Software Initiative (OSI)**

– non-profit corporation, founders: Todd Anderson, Chris Peterson, John "maddog" Hall, Larry Augustin, Sam Ockman, and Eric Raymond. Conceived in 1998. Not a membership organization. Currently there are five board members with Raymond as President.

- One-sentence sound bite: "Open source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code."
- Availability of source code
- ***www.opensource.org***

Free Software – according to FSF

- Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:
 1. The freedom to run the program, for any purpose (freedom 0).
 2. The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
 3. The freedom to redistribute copies so you can help your neighbor (freedom 2).
 4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.
- A program is free software if users have all of these freedoms.
- You should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.
- You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way.
- Regardless of how you got your copies, you always have the freedom to copy and change the software, even to sell copies. "Free software" does not mean "non-commercial".

Open Source Definition (OSD)

1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of the Author's Source Code
5. No Discrimination against Persons or Groups
6. No Discrimination against Fields of Endeavor
7. Distribution of License
8. License Must Not Be Specific to a Product
9. The License Must Not Restrict Other Software
10. The License must be technology-neutral

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or

other fee for such sale.



2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

CHAPTER 3: Licensing

Licensing or **grant license** means to get permission or authority. A licensor may grant a **license** under intellectual property laws to authorize a use (such as copying software or using an invention) to a licensee, sparing the licensee from a claim of infringement brought by the licensor. Licenses are valid for a particular length of time. This protects the licensor should the value of the license increase, or market conditions change. It also preserves enforceability by ensuring that no license extends beyond the term of the agreement.

Corporate Trademark and Brand Licensing

The licensing of company names, logos, or brands (referred to as corporate trademark/brand licensing) is one of the fastest-growing segments of the licensing business. Much of the growth is spurred by the fact that licensing provides enormous strategic, marketing and earning benefits to both licensor and licensee.

3.1 Types of licensing

1. Adding New License Type
2. Editing License Type
3. Deleting License Type

3.2 Commercial License versus Open Source License

What is commercial software?

- Software that is sold and support commercially
- (Open source software can be sold and support commercially too.)
- Perhaps “proprietary software” is more correct

Benefits of Commercial Software

- Benefits of providing access to commercial software in academic environment:
 - ✓ Understand the (business) model of software in commercial environment
 - ✓ Familiar with commercial software / solution
 - ✓ Comply with industrial standards

Benefits of OSS

- Availability of source code to study and experiment with
 - ✓ Beautiful source code to read
 - ✓ Do not have to re-invent the wheel
 - ✓ Free as in “freedom”
 - ✓ And sometimes: free as in “gratis”
- Does not depend on vendor.
 - ✓ Can choose vendor / support we like
 - ✓ Can fix bugs

Disadvantage of OSS

- If source code is not looked at, no need to have OSS
- There are also bad codes, unqualified persons
- Too much hype?

3.3 Open Source Software Licensing

An **open-source license** is a type of license for computer software and other products that allows the source code, blueprint or design to be used, modified and/or shared under defined terms and conditions. This allows end users to review and modify the source code, blueprint or design for their own customization, curiosity or troubleshooting needs. Open-source licensed software is mostly available free of charge, though this does not necessarily have to be the

case. Licenses which only permit non-commercial redistribution or modification of the source code for personal use only are generally not considered as open-source licenses.

Types of OSS licenses

- Copyright law: Must have permission to copy software
 - ✓ Permission is given by a license
 - ✓ Proprietary software: Pay for a license to use a copy/copies
 - ✓ OSS licenses grant more rights, but still conditional licenses
- Over 100 OSS licenses, but only a few widely used
- Can be grouped into three categories (differing goals):
 - ✓ Permissive: Can make proprietary versions (MIT, BSD-new)
 - ✓ Weakly protective: Can't distribute proprietary version *of this component*, but *can* link into larger proprietary work (LGPL)
 - ✓ Strongly protective: Can't distribute proprietary version *or* directly combine (link) into proprietary work (GPL)
- The most popular OSS licenses tend to be compatible

Compatible = you can create larger programs by combining software with different licenses (must obey all of them)

OSS licensing strategies

CHAPTER 4: Introduction on Web Technologies

Introduction on Web Technologies

World Wide Web, abbreviated as www and commonly known as the web is a system of interlinked hypertext documents accessed via the Internet. Web Technologies relates to the interface between web server and their clients. This information includes markup language, programming interfaces language and standards for document identification and display. The techniques used for the web development is generally known as web technologies.

Internet Concept

Internet is rich resources for the information and data, never ending, always changing and completely dynamic. Internet is a worldwide collection of interconnected computer network that used TCP/IP and its related services. Internet began in 1969 as a experimental four-computer network called ARPA net, which was designed by the U.S Defense Department., so that research scientist could communicate. After two year ARPA net grew to about two dozen sites. In 1990 ARPA net officially disbanded and the network which now consisted of hundreds of sites came to known as Internet.

The Internet consists of two types of computer called servers and clients

- Computers which offer information to be read are called servers.
- Computers that read the information offered called clients.

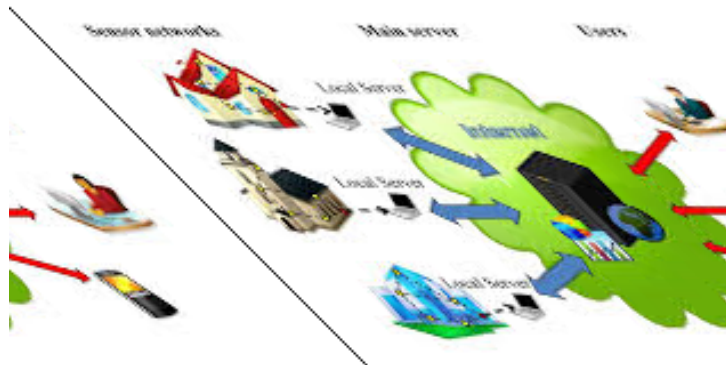


Fig: Showing the Wide Area Network

Server runs special Software (web server) that allows them to

- Respond to client request for information
- Accept data from clients.

Some of the most popular software which servers run to allow them to respond to client request for run to allow them to respond to client request for information are, Internet Information server (IIS), Apache web server, Microsoft personal web server

Client runs special software (Browser Software) that allows them to

- Locate the appropriate server
- Query the server for the information to be read. Example Internet explorer, Mozilla, Netscape, Google chrome, Safari

Communication on the Internet

TCP/IP is the only protocol used to send data all around the internet. TCP/IP is really two individual sections (TCP) a set of communication protocols and (IP) a unique address. Every machine connected to the Internet must have an address by which it can be located on the internet. This is called the IP address of the machine. No two machines can

have the same IP address. Hence each machine connected to the internet must have a unique IP address, which identifies that machine.

Internet is a worldwide network of network. It is important to have a governing body, which allocated unique IP address to organizations wishing to manage networks, which will be part of Internet.

A unique IP address therefore points to an actual computer connected via gateway to the Internet. This computer is known as Domain. i.e Place where information is available .This is a physical domain on the internet

Conceptually a server, which has a permanent IP address (i.e physical domain) can provide

- A gateway to other computers to access the internet AND/OR
- Provide information for the Internet clients to read AND/OR
- Provide a physical location on which server virtual Domain can be hosted.

When a websites provides internet clients information to read, the site is mounted as a virtual domain on a internet server, which is its (host) physical Domain.

Virtual Domains are identified by a name (eg. www.google.com, www.facebook.com)

Just like a physical Domain need to have unique IP address, so also Virtual Domain Names need to be unique on the internet.

What is a protocol?

- A protocol is a collection of rules and procedures for two computers to exchange information
- Protocol also defines the format of data that is being exchanged

What is TCP/IP?

- TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network.
- TCP stands for “Transmission Control Protocol”
- IP stands for “Internet Protocol”
- They are Transport layer and Network layer protocols respectively of the protocol suite
- The most well-known network that adopted TCP/IP is Internet – the biggest WAN in the world
- TCP is a connection-oriented protocol

Does not mean it has a physical connection between sender and receiver

TCP provides the function to allow a connection virtually exists – also called virtual circuit

- TCP provides the **functions**:

Dividing a chunk of data into segments

Reassembly segments into the original chunk

Provide further the functions such as reordering and data resend

- Offering a reliable byte-stream delivery service

Application
Presentation
Session

Transport
Network
Data Link
Physical
Application
Transport
Network
Network Interface

Fig: OSI Reference Model and TCP/IP Model

Application Layer

- Application layer protocols define the rules when implementing specific network applications
- Rely on the underlying layers to provide accurate and efficient data delivery
- **Typical protocols:**
 1. FTP – File Transfer Protocol - For file transfer
 2. Telnet – Remote terminal protocol - For remote login on any other computer on the network
 3. SMTP – Simple Mail Transfer Protocol - For mail transfer
 4. HTTP – Hypertext Transfer Protocol - For Web browsing
- TCP/IP is built on “connectionless” technology, each datagram finds its own way to its destination

Transport

- Transport Layer protocols define the rules of
 - ✓ Dividing a chunk of data into segments
 - ✓ Reassemble segments into the original chunk
- **Typical protocols:**
 - ✓ TCP – Transmission Control Protocol
 - Provide further the functions such as reordering and data resend
 - ✓ UDP – User Datagram Service
 - Use when the message to be sent fit exactly into a datagram
 - Use also when a more simplified data format is required

Network

- Network layer protocols define the rules of how to find the routes for a packet to the destination
- It only gives best effort delivery. Packets can be delayed, corrupted, lost, duplicated, out-of-order
- Typical protocols:
 - ✓ IP – Internet Protocol - Provide packet delivery

- ✓ ARP – Address Resolution Protocol - Define the procedures of network address / MAC address translation
- ✓ ICMP – Internet Control Message Protocol - Define the procedures of error message transfer

All the Virtual Domain Names must be registered with InterNIC

Registering a Virtual Domain with InterNIC

- Log on Inter NIC server.
- Fill Registration from online
- Pay the year registration fee to inter NIC.
- These register virtual domains with Inter NIC (Domain Name) virtual Domains will be scan its registered database to ensure that the domain name is unique.
- 30 day time is given to payment.
- If name is not unique, the registration is rejected.

Domain Name Extension

InterNIC root servers

.edu, .gov, .mil, .org, .com

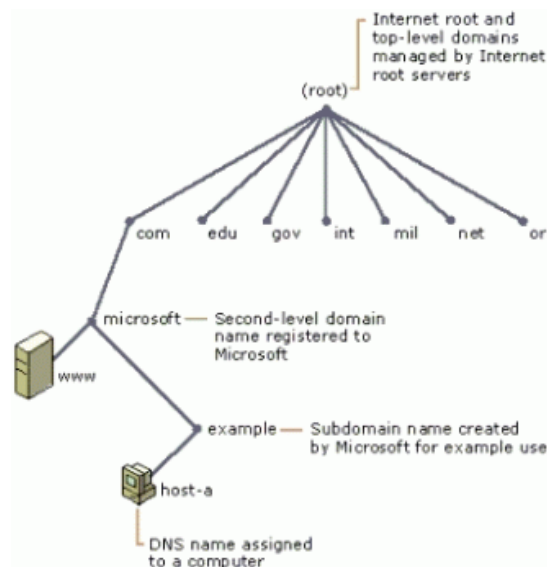
.edu – servers that provide educational services

.gov – server's that provide information about the government of country.

.mil – servers military information

.org – servers that provide info about organization in this world

.com – servers that provide commercial services on the internet



FTP

FTP is an important protocol for Internet business. File Transfer Protocol (FTP) is not just a protocol but also a services and as application. FTP is especially useful for transferring files between different computers. FTP provides the facility to transfer files between two computers running on different operating systems like windows and UNIX.

Software like cuteFTP, flash XP, Filezilla etc provides upload and download of files between the client computer and the server computer. It also performs various operations like deletes rename, change file permission, and so on and logout from the remote computer when done.

FTP as a service

FTP is a service for copying files from one computer to another. FTP as a protocol FTP is a protocol for copying files between two computers. The client and the server applications both use it for communication to ensure that the new copy of the file is identical to the original.

Server and Web Server

The Computer serving information from a central location is called server. Which location is called server. This stores web pages in form of direction and files. Web Pages are created using html syntax. There pages must be organized and stored at a central computer.

The server computer run special software called web server that allows.

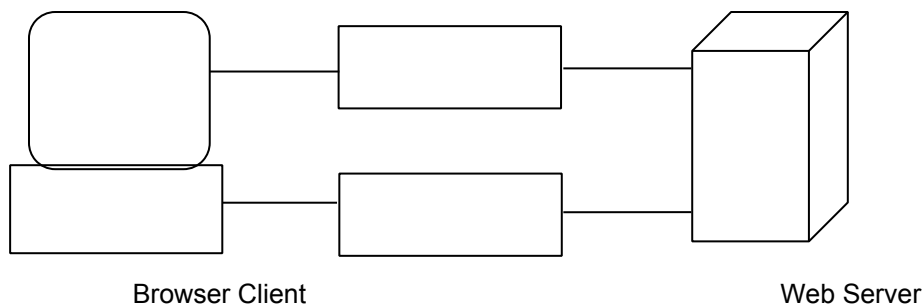
- Website Management
- Accept a client's request for information
- Respond to a client request by providing the page with the required information.

Web Client/ Browser

Computer that offers the facility to read information stored in web pages are called clients. Web client runs special software called a browser that allows them to :

- Connect to an appropriate server
- Query the server for the information to be read
- Provides as interface to read the information returned by the server. eg Internet explorer, Netscape, Fire-Fox.

Understanding how a browser communicates with a web server



Establish connection

Before a client and server can exchange information they must first establish a connection. TCP/IP is used to let computers establish a link between a web server and web browser over the internet. To communicate with web

server, the client machine must sub protocol that must be used i.e HTTP, FTP etc. The client browser will attempt to locate the server based on the IP address supplied and established a connection.

Client Issues a Request and Server sends a Response

When a Browser connects to a web server using an appropriate protocol name, IP address and port number and the web server treats this connection to be a request for the 'Default web document' The web server then dispatches the 'Default web document' to the client who connected.

If the client requires viewing any other web page then the client can specify the web page name along with the connection information. Thus the complete connection and web pages information will now be specified as.

Protocol:// server name: port number/ web pages name.

Server terminates the connection

It is the server's responsibility to terminate the TCP/IP connection with the Browser after it responds to Browser's request. If user clicks on the Browser's Stop button, the Browser must close the connection

Also when computer crash by either a Browser or a web server must be recognized by the surviving computer, which in true, will close the connection.

URL: full form of URL is Uniform Resource locator. It is the location of a web site or called Domain name of the site.

WWW (World Wide Web) use HTTP (Hyper Text Transfer protocol) to make communication possible between web server and web browser.

CHAPTER 5: Web Development with HTML & DHTML (6 Hrs)

HTML (Hyper Text markup language):

The language used to develop web page is called HTML and is interpreted by a browser. The HTML is devised from GML (Generalized Markup Language). GML was developed by Charles Goldfrab, Edward Moster and Raymond Lorie at IBM. Later American National Standards Institute (ANSI) took the basics of GML

HTML Tag: Tags are instructions that are embedded directly into the text of the document. HTML tags begin with an open angle bracket (<) and end with a close angle bracket (>).

Paired tags: A tag is said to be a paired tag if it, along with a companion tag, flanks the text with its companion text. i.e. it has got an opening tag and closing tags the effect is seen in between them. e.g tag is paired tag. The tag with its companion tag causes the text contained between them to be bold... is called opening tag and is called closing tag.

Bold Text

<I>Italic Text</I>

<u>Underline Text</U>

Singular tag: A stand-alone or singular tag does not have a companion tag. Eg
 tag will insert a line break. This has no any companion tag.

<hr/> draw the horizontal line

Structure of HTML program:

<HTML>

 <HEAD>

 <TITLE>

 </TITLE>

 </HEAD>

 <BODY>

 </BODY>

</HTML>

The entire webpage is enclosed within <HTML> tags within these two tags two distinct sections are created using the <HEAD> </HEAD> and <BODY> </BODY> tags. Information placed in head section is essential to the inner working of the document and has nothing to do with the content of document. The information placed within the <HEAD> </HEAD> tags is not displayed in the browser. The information written within the <BODY> </BODY> is displayed in the browser. Background color, text color, font size etc can be specified as attributes of <BODY> tag.

Program #1

<HTML>

 <HEAD>

```
<TITLE>~::~ Official web Sites Of KhEC ~::~</TITLE>
```

```
</HEAD>
```

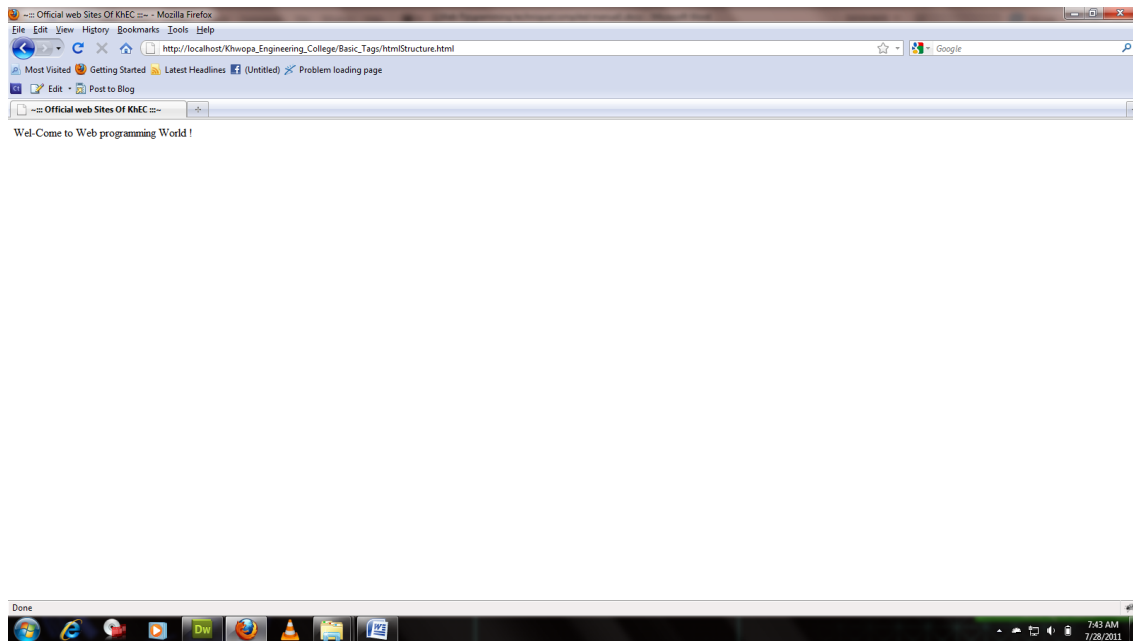
```
<BODY>
```

```
Wel-Come to Web programming World !!!
```

```
</BODY>
```

```
</HTML>
```

Output:



Text Formatting:

<p>: used for paragraphs on encountering this tag, the browser moves onto a new line skipping one line between the previous line and the new line.

**
:** line break start from a new line and not continue on the same line.

Heading styles: HTML supports six different levels of headings. The highest-level is **<H1>**. The size varies from **<H1>** to **<H6>**.

Example:

```
<H1>Heading 1</H1>
```

```
<H2>Heading 2</H2>
```

```
<H3>Heading 3</H3>
```

`<H4>Heading 4</H4>`

`<H5>Heading 5</H5>`

`<H6>Heading 6</H6>`

Output:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

<hr> tag: This tag draws a horizontal line across the whole page, whenever specified. These attributes are: **Align:** Align the line on the browser screen which is by default, aligned to the center of the screen.

Align = left

Align = right

Align= center

Size: Changes the size of the rule.

Width: Sets the width of the rule.

Example:

`<hr align = left width =10 size =4>`

<I> tag: This tag displays text in Italics.

Example:

`<I> KhEC </I>`

<u> tag: This tag displays text as underlined. And is used as `<u>....</u>`

Example:

`<u> KhEC </u>`

** tag:** This tag displays text as Bold. And is used as `....`

Example:

` KhEC `

<Center> tag: This tag is used to center everything found between `<center>.....</center>` tag like text, list, images, tables or any other page element.

Example:

`<center> welcome to KHWOPA </center>`

** tag:**

All text specified within the tags and will appear in the font, sizes and color as specified as attributes of the tag . The attributes are:

FACE: Sets the font to the specified font name.

SIZE: Sets the size of the text and value is between 1 to 7.

Color: Sets the color of the text.

Example:

```
<FONT FACE= "Comic Sans Ms" size = 6 color = RED> Khwopa Engineering College </FONT>
```

Setting background color:

Bgcolor = "Name of the color"

```
<body bgcolor ="RED">
```

....

```
</body>
```

Lists:

There are two types of list:

1. Unordered list (bullets): An unordered list starts with the tag and ends with . Each list item starts with the tag . The attributes are specified within tag.

Attributes:

Type: Specifies the type of the bullet.

Type= Fill round (solid round block)

Type= Square (solid square black bullet)

Eg:

```
<UL type="square">
```

```
    <LI>B.E Architecture</LI>
```

```
    <LI>B.E Computer</LI>
```

```
    <LI> B.E Civil</LI>
```

```
    <LI> B.E Electronics & communication </LI>
```

```
</UL>
```

Output:

- B.E Architecture
- B.E Computer
- B.E Civil
- B.E Electronics & communication

2. Ordered lists (Numbering): An ordered list start with the tag and ends with . Each list item starts with tag . The attributes that can be specified within are:

Type: Controls the numbering scheme.

Type= "1" (count number 1, 2, 3....)

Type= "A" (give uppercase letters A, B, C,...)

Type= "a" (give lowercase letters a, b, c,...)

Type= "I" (give uppercase Roman letter I, II, III,...)

Type= "i" (give lowercase Roman letter i, ii, iii,...)

Start: Sets any numeric value.

Example:

```
<OL type = "I" start="5">
```

```
  <LI> Khwopa Engineering College</LI>
```

```
  <LI> Kantipur City College </LI>
```

```
  <LI value="9"> Acme Engineering College</LI>
```

```
  <LI> Himalayan White House Engineering College </LI>
```

```
</OL>
```

Output:

5. Khwopa Engineering College
6. Kantipur City College
9. Acme Engineering College
10. Himalayan White House Engineering College

3. Definition List: Definition list appears within tags <DL> and </DL>. It consists of two parts:

- Definition terms: appears after tag <DT>
- Definition description: appears after the tag <DD>

Example:

```
<DL>
```

```
  <DT> Mouse</DT>
```

```
  <DD> Keyboard</DD>
```

```
  <DT> Tablet</DT>
```

```
  <DD> Touch Panel</DD>
```

```
</DL>
```

Output:

Mouse

Keyboard

Tablet

Touch Panel

Q. Practice the following condition.

COMPUTER SHOP

1. Desktop Computer
 - Pentium I
 - Pentium II
 - Pentium III
 - Pentium IV

2. Laptop

- HP
 - HP with i3 Processor
 - HP with i5 Processor
 - HP with i7 Processor
- DELL
 - DELL STUDIO
 - DELL INSPIRION
- TOSHIBA
- MSI
- ACCER
- BENEQ
- SONI VAIO

Adding graphics to HTML Document:

HTML allows static and animated images in an HTML page. It accepts two pictures file formats .gif and .jpg. Image can be inserted into a web page using the tag , along with the name of the image file (filename. Gif or file name.jpg or file.jpeg). The attributes of the tag are:

Align: TOP: indicates the text after the image to be written at the top, next to the image.

Align= Middle: indicates the text after the image to be written at the middle, next to the image.

Align= center

Align= Right

Border: Specifies the size of the border of image.

Width: Specifies the width of the image.

Height: Specifies the width of the image.

Hspace: indicates the amount of space to the left and right of the screen.

Vspace: amount of the space to the top and bottom of the image.

Alt: Displays the text incase the browser each unable to display the image specified in the SRC attribute.

Syntax:

```
<IMG width = 400 height = 50 Border= 0 Hspace=0 SRC = "grsuwal.jpg" align = "center" />
```

Example:

```
<HTML>
```

```
  <head>
```

```
    <title> .....</title>
```

```
  </head>
```

```
  <body>
```

```
    <B> way for inserting image</B>
```

```
    <center>
```

```
      <I> Image without using attributes</I>
```

```
      <Image SRC ="image1.gif"><br />
```

```
    </Image using attributes</I>
```

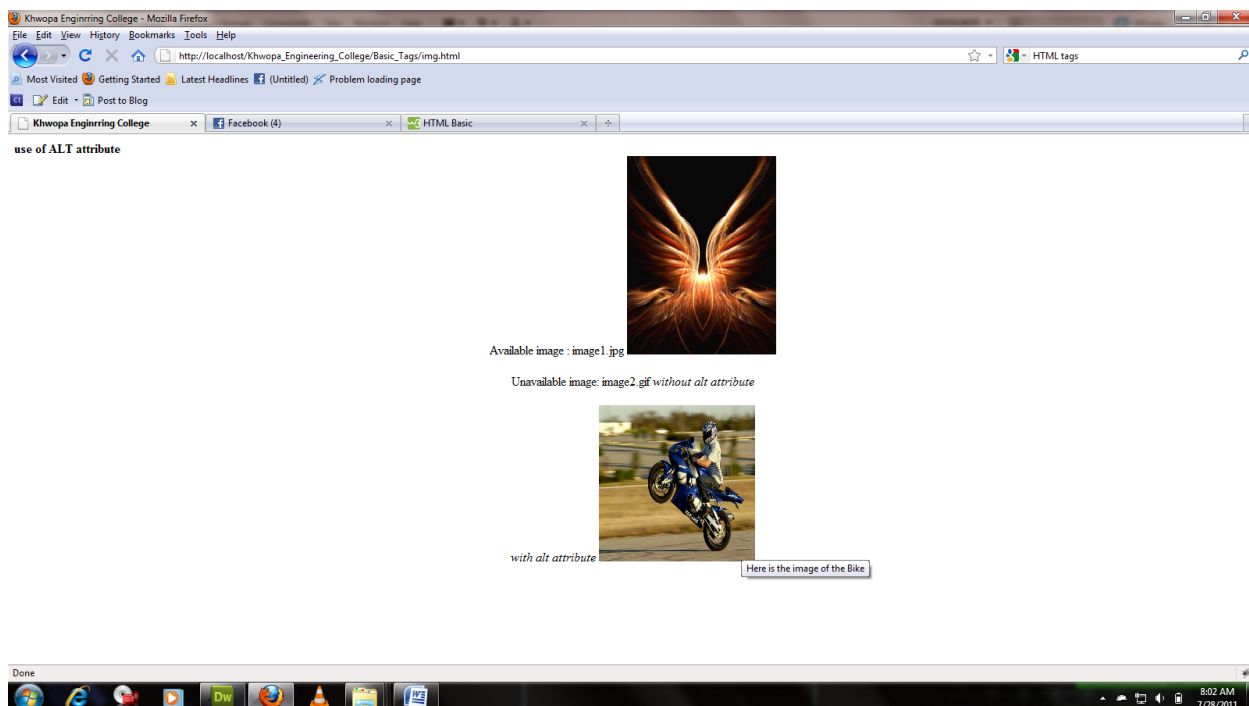


```
<Image border=3 height=200 width=200 SRC="khwopa.jpg">
</center>
</body>
</html>
```

Using ALT attributes

```
<html>
<head>
<title>Khwopa Engineering College</title>
</head>
<body>
<B>use of ALT attribute</B>
<center> Available image : image1.jpg
<img src = "../image1.jpg" width="191" height="254"><br /><br />
Unavailable image: image2.gif
<I> without alt attribute </I>
<br /><br />
<I> with alt attribute </I>
<img src = "../pulsar_bike.jpg" width="200" height="200" alt = "Here is the image of the Bike" title = "Here is
the image of the Bike"><br />
</center>
</body>
</html>
```

Output:



EMBED Tags

<EMBED ...> puts a browser plug-in in the page. A *plug-in* is a special program located on the client computer (i.e. not on your web server) that handles its own special type of data file. The most common plug-in are for sounds and movies. The <EMBED ...> tag gives the location of a data file that the plug-in should handle.

In its simplest use, <EMBED ...> uses the SRC attribute to indicate the location of the plug-in data files, and usually also gives a WIDTH and HEIGHT of the plug-in area. For example, the following code embeds a MIDI file of the *1812 Overture* in the page:

Example:

```
<embed src="../../../SUMMER OF 69(LIVE).MPG" width="200" height="200" align="middle">
```

Tables:

A table is a two dimensional matrix consisting of rows and columns. All table related tags are included between the <table> and </table> tags. Each row of a table is described between the <tr></TR> tags and column between <TD></TD> tags.

Rows can be of two types.

1. Header rows:

It is defined using <TH> </TH> tags. The content of a table header row is automatically centered and appear in boldface.

2. Data rows:

Data cell placed in the horizontal plane creates a data row. Data cell hold data that must be displayed in the table. It is defined using <tr> </TR> tags. Attributes included in the <table> tag are:

Align: Indicates horizontal alignment. It can be set to left, center or right. **Valign:** Indicates vertical alignment. It can be set to top, middle or bottom. **Width:** Sets the width to a specific no. of pixels. If width is not specified, the data cell adjusted based on cell data value. **Border:** Indicates the border thickness of the table.

Cellpadding: Specifies the distance between the data in a cell and the boundaries of the cell.

Cellspacing: Controls the spacing between adjacent cells.

Colspan: This attributes is used inside <TH> or <TD> tag. This attribute is useful when one row of the table needs to be a certain number of columns wide.

Rowspan: This attribute is useful when one column needs certain no. of rows.

Caption Tag:

Table heading are called captions. It is written as <caption> </caption>. This paired tag appears within the <table> </table> tags. Its attributes are:

Align: Align= bottom: Place the caption below the table.

Align: top: Place the caption above the table.

Example:

Programs using width and border of a table.

<HTML>

```
<head>
    <title>The table attributes</title>
</head>
<body>
    <center>
        <table border=5 width=30%>
            <caption align="top">
                <B>personal Information </B>
            </caption>
            <tr >
                <TH>Subject: </TH>
                <TH>Credit Hours:</TH>
            </TR>
            <tr align=center>
                <TD> Web programming Technique </TD>
                <TD>4</TD>
            </TR>
            <tr align=center>
                <td>Computer Graphics</td>
                <td>4</td>
            <tr >
                <tr align=center>
                    <td>C-Programming</td>
                    <td>3</td>
                <tr >
            </table>
        </center>
    </body>
</HTML>
```

output:

personal Information

Subject:	Credit Hours:
Web programming Technique	4
Computer Graphics	4
C-Programming	3

// Cellpadding and Cellspacing attribute:

```
<HTML>
    <head>
        <title>The table attributes</title>
```

```
</head>
<body bgcolor="#CCCCCC">
  <center>
    <table border=5 width=30% cellpadding="5" cellspacing="2">
      <caption align="top">
        <B>Cell Padding And Cell Spacing</B>
      </caption>
      <tr >
        <TH>Subject: </TH>
        <TH>Credit Hours:</TH>
      </TR>
      <tr align=center>
        <TD> Web programming Technique </TD>
        <TD>4</TD>
      </TR>
      <tr align=center>
        <td>Computer Graphics</td>
        <td>4</td>
      <tr >
        <tr align=center>
          <td>C-Programming</td>
          <td>3</td>
        <tr >
      </table>
    </center>
  </body>
</HTML>
```

Cell Padding And Cell Spacing

Subject:	Credit Hours:
Web programming Technique	4
Computer Graphics	4
C-Programming	3

// Using bgcolor attribute

```
<html>
  <head>
    <title>bgcolor attributes </title>
  </head>
  <body bgcolor = lightgrey>
```

```
<center>
    <strong> People Information</strong>
    <table border =2 width=50% align=center>
        <tr >
            <th bgcolor=grey>Name </th>
            <th bgcolor=grey>Address </th>
        </tr>
        <tr align= center>
            <td bgcolor =black><font color =white>Er. Ganesh Ram Suwal</font></td>
            <td bgcolor=violet><font color="Red">Bhaktapur</font></td>
        </tr>
        <tr align=center>
            <td bgcolor=gray>Er. Bikash Chawal</td>
            <td bgcolor = yellow>Bhaktapur</td>
        </tr>
    </table>
</center>
</body>
</html>
```

People Information

Name	Address
Er. Ganesh Ram Suwal	Bhaktapur
Er. Bikash Chawal	Bhaktapur

// Using colspan & Rowspan attributes.

```
<html>
    <head>
        <title>bgcolor attributes </title>
    </head>
    <body bgcolor = lightgrey>
        <center>
            <strong> Khwopa Engineering College (Class Routine ) </strong>
            <table width="100%" border="1" cellspacing="1" cellpadding="3">
                <tr >
                    <td rowspan="2"><strong>Day</strong></td>
                    <td><strong>1<sup>st</sup></strong></td>
                    <td><strong>2<sup>nd</sup></strong></td>
                    <td><strong>3<sup>rd</sup></strong></td>
                    <td><strong>4<sup>th</sup></strong></td>
                    <td><strong>5<sup>th</sup></strong></td>
```

```
<td><strong>6<sup>th</sup></strong></td>
<td><strong>7<sup>th</sup></strong></td>
<td><strong>8<sup>th</sup></strong></td>
</tr>
<tr >
<td><strong>7:15-8:05</strong></td>
<td><strong>8:05-8:55</strong></td>
<td><strong>8:55-9:45</strong></td>
<td><strong>9:45-10:35</strong></td>
<td><strong>10:35-11:25</strong></td>
<td><strong>11:25-12:15</strong></td>
<td><strong>12:15-1:05</strong></td>
<td><strong>1:05-1:55</strong></td>
</tr>
<tr >
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr >
<td>Monday</td>
<td colspan="2">Web programming lecturer</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td colspan="3">Web programming Lab (Grp A)</td>
</tr>
<tr >
<td>Tuesday</td>
<td colspan="3">Web programming Lab (Grp B)</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
```

```
<tr >

    <td>Wednesday</td>
    <td>&nbsp;</td>
    <td colspan="2">Web programming lecturer</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>

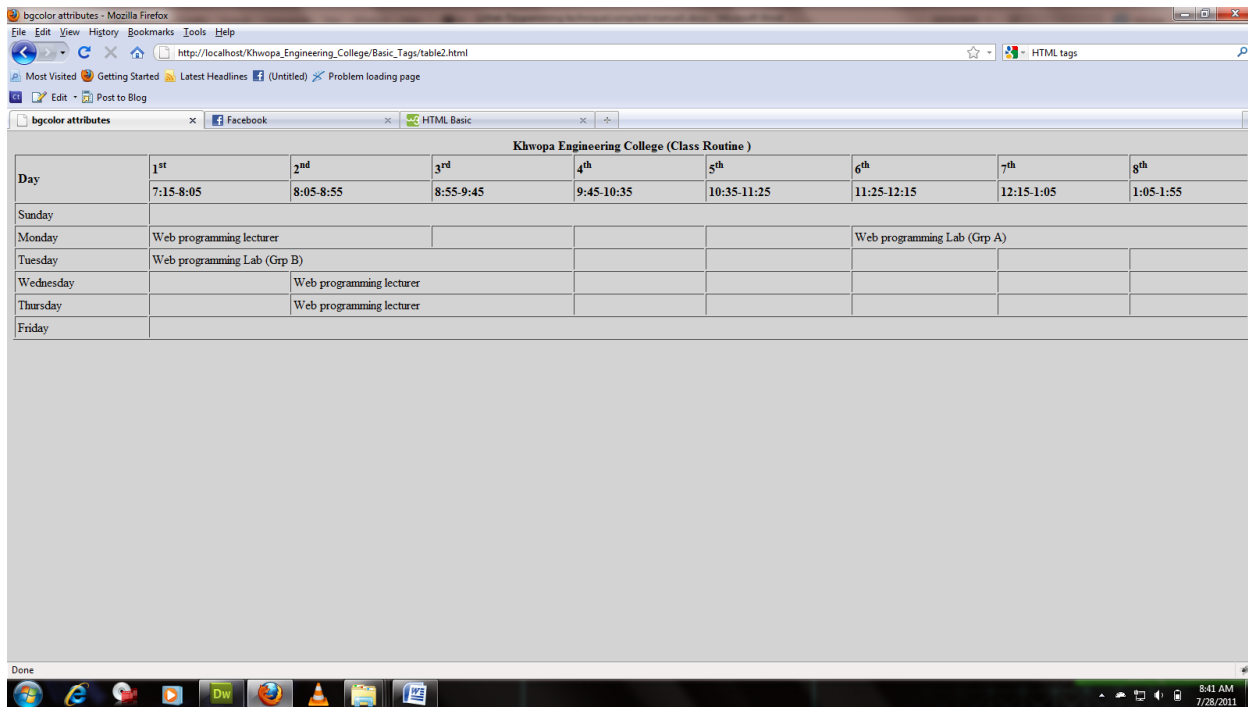
</tr>
<tr >

    <td>Thursday</td>
    <td>&nbsp;</td>
    <td colspan="2">Web programming lecturer</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>

</tr>
<tr >

    <td>Friday</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
    <td>&nbsp;</td>

</tr>
</table>
</center>
</body>
</html>
```



Day	1 st	2 nd	3 rd	4 th	5 th	6 th	7 th	8 th
	7:15-8:05	8:05-8:55	8:55-9:45	9:45-10:35	10:35-11:25	11:25-12:15	12:15-1:05	1:05-1:55
Sunday								
Monday	Web programming lecturer					Web programming Lab (Grp A)		
Tuesday	Web programming Lab (Grp B)							
Wednesday		Web programming lecturer						
Thursday		Web programming lecturer						
Friday								

Linking Documents:

Links: HTML allows linking to other html documents as well as images. Clicking on a section of text or on image in one web page will open an entire web page or an image. The text or an image that provides such linkages is called hypertext, a hyperlink or a hotspot. The browser distinguishes hyperlinks from normal text. Every hyperlink,

- Appears *blue in color* (by default but can be changed by html program).
- The hyperlink text/image is *underlined*.
- If the hyperlink is used in an image the border of blue color in an image is appeared
- When the mouse is placed over it, the mouse cursor changes to the shape of a hand. Links are crated in a web page by using the <A> tags. Any things written between the <A> tags becomes a hyperlink/hotspot. The document to be navigated needs to be specified. By using HREF attribute of the <A> tag the next web page or image can be specified.

Syntax:

Hyperlinks can be of two types.

1. External Document References.
2. Internal Document References.

External document references:

Example:

`KhEC` Here KhEC becomes a hyperlink & links to another document, KhEC.html. The HTML file must be present in the current working directory. If the file is not present in the current directory, a relative or absolute path can be specified.

Anchors:

By default a hyperlink takes users to the beginning of the new web page. It might be necessary to jump to a particular location within the new web page. To enable a jump to a specific location on a web page, anchors can be set up. Anchors target hyperlink a specific location point on a web page. It is summarized in two steps.

Step: 1

Mark the location to be jumped to.

Syntax:

``

Example:

`< A name= "Point1">`

Here, location to be jumped is point1.

Step: 2

While jumping to a specific web page & a specific location on the web page, we require page name along with the name of the location to be jumped on that page.

Syntax:

``

Example:

` KhEC `

Internal Document references:

This is used when a jump is required to a different location in the same document. To perform the link we again follow two steps. First identify a location with a name & then jump to that location using the name.

Syntax:

`< A Name = "location_name">`

``

Here the absence of filename.htm before the #symbol indicates a jump is required within the same document.

Example:

``

`KhEC `

Example:

// hyper linking to a HTML file.

`< html>`

`<head><title>Hyperlink </title></head>`

`<body bgcolor = "gray">`

```
<center>
    <H1> KhEC </H1><br />
    <img width = 400 height = 50 src = "college.gif">
</center>
<HR>
<H4> KhEC provides the following courses for undergraduate Program : </H4>
<UL>
    <LI> <A href = "Computer.htm"> B.E computer </A></LI>
    <LI> <A href = Electronics.htm">B.E Elex & Comm. </A></LI>
    <LI> <A href = "civil.htm">B.E civil </A></LI>
    <LI> <A href = "Architecture.htm"> B.Arch </A></LI>
</UL>
<Spacer size = 200 > click for more details!
</body>
```

```
</html>
```

[Note: Save it as index.htm]

Code for Computer.htm

```
<html>
    <body background= "University.gif">
    <Marquee> KhEC </marquee>
    <center>We cover following Subject in B.E computer for different semester. </center>
    <br />
    <ol>
        <li>1st Semester </li>
        <ul>
            <li> C programming </li>
            <li> Math- I </li>
            <li>Engg. Drawing</li>
            <li>Workshop Technology </li>
            <li>Physics </li>
            <li>Computer concept </li>
        </ul>
        <li> 2nd Semester </li>
        <ul>
            <li>Object oriented programming </li>
            <li>Maths II </li>
            <li>Applied Mechanics </li>
            <li>Chemistry</li>
            <li>Electrical Material </li>
        </ul>
        .....
    </ol>
```

Continue yourself till 8th semester.

.....

</body>

</html>

Linking particular Location in a separate document:

Code for index2.htm

<html>

<body bgcolor = "gray">

<centre><h1><Marquee>Purbanchal University </marquee></h1></centre>

BE computer

< a href = "Stream.htm#Elx&com">BE Elx & com

B.Arch

 BCA

 BIT

</body>

</html>

Code for stream.htm:

<html>

<body bgcolor = "gray">

<h2> B.E computer syllabus </h2>

 First semester

C-programming

Math

Communication Technique

physics

Computer concept.

.....

Similarly repeat upto 8th semester.

<h2>B.E elx & communication Syllabus </h2>

First semester

C-programming.


```
</ul>
<ul>
    .....
</ul>
<ul>
    <li>7th semester
    <li>Web- Technology.
    <li>Antenna
    <li> Org. & Mgmt.
    <li> Elective I.
    .....
</ul>
<ul>
    <li>8th semester
    .....
</ul>
</body>
</html>
```

Image as hyperlinks:

An image can be made a hotspot by enclosing an tag within <A> tags. The tag places the image on the screen, & because the tag is enclosed within the <A> tags, becomes a hotspot.

Syntax:

```
<A href = "filename.htm"><img src = "imgname.gif"></A>
```

Here the picture image acts as hotspot and navigates to a file filename.htm.

Image Maps:

When a hyperlink is created on an image, clicking on any part of the image will lead to opening of the document specified in the <A href ...> tag. In order to link multiple documents to the same image, the image is divided into multiple sections and allows lining of each section to a different document. This technique is called image maps. Image maps can be created and applied to an image so that specific portion of image can have linked to a different file/image.

Creating an image map is in two-step process.

Step: 1

Create an image map i.e divide the image into various areas.

This is done using the <MAP></MAP> tags.

Syntax:

```
<MAP NAME = "map name">
```

Within the <MAP> </MAP> tags the <AREA> tag is specified. This tag defines specified. This tag defines specific region and take the attributes:

Shape: The shape of a region can be rect, circle, polygon, default.

Coords: Defines the coordinates to different shapes.

Rectangle – x1, y1, x2, y2.

Circle - centerx, centery, radius.

Polygon - 3 or more pairs of coordinates.

Default- no coordinate is specified.

href: Takes the name of the .htm file that is linked to the particular area on the image.

Example:

```
<MAP name= "test">
    <Area Shape = "rect" Coords= "52,65,122,83" href= "first.htm">
    <Area Shape= "rect" Coords = "148,65,217,89" href ="second.htm">
</MAP>
```

Step: 2

It deals with the particular part of image. For this, the takes an attribute called USEMAP that takes the name of the image map as value. This value is preceded with # sign.

Syntax:

```
<img Usemap= "#map_name">
```

Example:

```
<img src = abc.gif "usemap = "#test">
<HTML>
    <head> <title> .....</title></head>
    <body bgcolor = "gray">
        <map name = "alert_map">
            <Area.shape = "Rect" coords = "102,74,164,94" href ="strem.htm">
            <area shape = "rect coords = "209, 74,272,96" href ="missing.htm">
        </map>
        <center><img src = "alert.gif" Use map = "#alert_map"></center>
    </body>
</html>
```

Code for stream.htm:

```
<html>
    <head><title> .....</title></head>
    <body>
        <h2> Streams run by PU </h2>
        <br /> <i> which stream do you want to visit?</i>
        <br />
        <ul>
            <li> <a href = "comp.htm">B.E comp </a>
            <li> <a href = "Elx and comm.htm">B.E Elx and comm. </a>
            <li><a href = "civil.htm">B.E Civil </a>
        </ul>
```

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

Code for missing.htm:

```
<html>
  <head><title> .....</title></head>
  <body>
    <h2> you are missing out on valuable information! </h2>
  </body>
</html>
```

Frames:

The HTML tags that divides a browser screen into two or more HTML recognizable unique regions is the <Frameset></Frameset> tags. Each unique region is called a frame. Each frame can be loaded with a different document and hence, allow multiple HTML documents to be seen concurrently. The <Frameset> & </Frameset> tags are embedded into the HTML document. These tags require one of the following two attributes.

ROWS:

This attributes is used to divide the screen into multiple rows. Depending on the required size of each row value can be

- Number of pixels
- Expressed as percentage on the screen resolution.
- The symbol *, indicating the remaining space.

Cols:

This attribute is used to divide the screen into multiple columns.

Example:

```
<frameset rows = "33%, 33%, 33%"> - Divide the browser screen into 3 equal horizontal sections.</frameset>
<frameset cols = "50%, 50%"> - Splits the 1st horizontal section into 2 equal vertical sections.</frameset>
<frameset cols = "50%, 50%"> - Splits the 2nd horizontal section into 2 equal vertical sections.</frameset>
```

<Frame> Tag:

Once the browser screen is divided into rows and columns, each unique section can be loaded with different HTML documents. This is achieved by using the <Frame>tag. Which consist of the following attributes?

Src = "url" – indicates URL of the document to be loaded into the frame.

Marginheight = "n" – Specifies the amount of white space to be left at the top and bottom of the frame.

Margin width = "n" – Specifies the amount of white space to be left along the sides of the frame.

Name: "name" → gives unique name to the frame so it can be targeted by other documents.

Noresize – Disables the frames resizing capability.

Scrolling -- Controls the appearance of horizontal and vertical scrollbars in a frame. This takes values yes/No/Auto.

Example:

```
<frameset rows = "30%",*>
    <frameset cols = "50%, 50%">
        <frame src = "file1.htm"> -- Loads the 1st frame with "file1.htm"</frame>
        <frame src= "file2.htm"> -- Loads 2nd frame</frame>
    </frameset>
    <frameset cols = "50%, 50%">
        <frame src = "file3.htm"> </frame>
        <frame src = "file4.htm"> </frame>
    </frameset>
</frameset>
```

Targeting Named Frames:

Whenever a hyperlink, which loads a document in a frame, is created, the file referenced in the hyperlink will be opened and will replace the current document that is in the frame. This is done by using the Name attribute of the <frame> </Frame> tags. The Name takes one parameter, which is its frame name. The attributes, via which the frame name is specified is the TARGET attribute. TARGET = "filename". &, The attribute via which the HTML file name is specified is the HREF attribute which is a part of <A> tag & is given by

```
<A href = "index.htm" TARGET = "main"> click here</A>
```

Example: Frame identification:

```
<Frameset cols = 30%, 70%>
    <Frame name = "part"> </frame>
    <frame name = "main"> </frame>
</frameset>
```

Hyperlink Specification:

```
<A href = "index.htm" Target = "main"> click here </A>
```

Here an index.htm is loaded into the frame named "man" when the hyperlink "click here" is clicked.

Code for frames.htm

```
<html>
    <frameset rows = "70, *">
        <frame src = "header.htm" marginheight = 0 marginwidht = 0 name = "frame1"> </frame>
        <frameset cols = "35%, *">
            <frame src = "pu.htm" name = "frame2">
            <frame src = "desc.htm" name = "frame3">
        </frameset>
    </frameset>
</html>
```

Code for header.htm

```
<html>
    <body bgcolor = "gray">
```

```
        <centre><h1>Purbanchal University </h1></centre>
    </body>
</html>
```

Code for pu.htm

```
<html>
    <body bgcolor = "yellow">
        <h4> constituent college of pu </h4>
        <ul>
            <li>PUSET</li>
            <li>Management college. </li>
            <li>CPAD</li>
        </ul>
    </body>
</html>
```

Code for desc.htm

```
<html>
    <body bgcolor = "red">
        <h3> programs run by PUSET are
        <ul>
            <li> B.E computer.</li>
            <li>B.E Elx & comm.. </li>
            <li>BCA</li>
            <li>MCA</li>
            <li>BIT</li>
        </ul>
        <br /> <h3> programs run by Management college are:
        <ul>
            <li>BBA</li>
            <li>MBA</li>
        </ul>
        <br />
        <h3>programs run by CPAD are:
        <ul>
            <li>Master in Population science. </li>
            <li>Master in Environmental Science. </li>
        </ul>
    </body>
</html>
```


Code for pu.htm:

```
<html>
  <frameset rows = "70, *">
    <frame src = "header.htm" marginheight = 0 name = "header"> </frame>
    <frameset cols = "25%, *">
      <frame name = "index" src = "indx.htm"> </frame>
      <frame name = "detail" src = "intro.htm"> </frame>
    </frameset>
  </frameset>
</html>
```

Code for header.htm:

```
<html>
  <body bgcolor = "red">
    <h1><centre> Purbanchal University </centre> </h1>
    <img src = "logo.gif" align = middle>
  </body>
</html>
```

Code for index.htm:

```
<html>
  <body background = "pu.img">
    <h4> constituent colleges of PU
    <ol>
      <li> < a href = "puset.htm" Target = "Details">PUSET </a></li>
      <li><a href = "management.htm" Target= "Details">Management college </a></li>
      <li><a href = "CPAD.htm" Target= "Details">CPAD</a></li>
    </ol>
  </body>
</html>
```

Code for intro.htm:

```
<html>
  <body bgcolor = "gray">
    <p><i> Purbanchal university is established in 2056. It has started several technical programs with a view to introduce New technology in Nepal. The university has its 3 constituent colleges all location at biratnagar...</i></p>
  </body>
</html>
```

Constituent college of PU.

1. PUSET

2. Management College

3. CPAD

Purbanchal University is established in 2056. It has started several technical programs to introduce new technology in Nepal. The university has its 3 constituent college.....

Purbanchal University

Again:

- write code for PUSET.htm
- Write code for management.htm
- Write code for cpad.htm.

HTML FORM

`<form>` is a pair tag. HTML forms are used to create (rather primitive) GUIs on Web pages.

- Usually the purpose is to ask the user for information
- The information is then sent back to the server

A form is an area that can contain form elements

- The syntax is: `<form parameters> ...form elements... </form>`
- Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
 - ✓ Other kinds of HTML tags can be mixed in with the form elements
- A form usually contains a Submit button to send the information in the form elements to the server
- The form's *parameters* tell JavaScript how to send the information to the server (there are two different ways it could be sent)
- Forms can be used for other things, such as a GUI for simple programs

The `<form arguments> ... </form>` tag encloses form elements (and probably other HTML as well)

- ✓ The arguments to form tell what to do with the user input

action="url" (required)

Specifies where to send the data when the Submit button is clicked

method="get" (default)

Form data is sent as a URL with `?form_data` info appended to the end

Can be used *only* if data is all ASCII and not more than 100 characters

method="post"

Form data is sent in the body of the URL request

Cannot be bookmarked by most browsers

target="target"

Tells where to open the page sent as a result of the request

target=_blank means open in a new window

target=_top means use the same window

INPUT TAG `<input>`

Most, but not all, form elements use the input tag, with a type="..." argument to tell which kind of element it is type can be text, checkbox, radio, password, hidden, submit, reset, button, file, or image

Other common input tag arguments include:

name: the name of the element

value: the "value" of the element; used in different ways for different values of type

readonly: the value cannot be changed

disabled: the user can't do anything with this element

Other arguments are defined for the input tag but have meaning only for certain values of type

1. text field:

```
<input type="text" name="textfield" value="with an initial value">
```

A text field:

2. multi-line text field

```
<textarea name="textarea" cols="24" rows="2">Hello</textarea>
```

A multi-line text field:

3. password field:

```
<input type="password" name="textfield3" value="secret">
```

A password field:

4. submit button:

```
<input type="submit" name="Submit" value="Submit">
```

5. reset button:

```
<input type="reset" name="Submit2" value="Reset">
```

6. plain button:

```
<input type="button" name="Submit3" value="Push Me">
```

A submit button:

A reset button:

A plain button:

6.1. submit: send data

6.2. reset: restore all form elements to their initial state

6.3. button: take some action as specified by JavaScript

7. checkbox:

```
<input type="checkbox" name="checkbox" value="checkbox" checked>
```

A checkbox: ☒

type: "checkbox"

name: used to reference this form element from JavaScript

value: value to be returned when element is checked

Note that there is *no text* associated with the checkbox—you have to supply text in the surrounding HTML

8. Radio buttons:

```
<input type="radio" name="radiobutton" value="myValue1">male<br>
```

```
<input type="radio" name="radiobutton" value="myValue2" checked>female
```

Radio buttons:

☐ male

☒ female

If two or more radio buttons have the same name, the user can only select one of them at a time

This is how you make a radio button “group”

If you ask for the value of that name, you will get the value specified for the selected radio button

As with checkboxes, radio buttons do not contain any text

9. menu or list:

```
<select name="select">
```

```
<option value="red">red</option>
```

```
<option value="green">green</option>
```

```
<option value="BLUE">blue</option>
```

```
</select>
```

A menu or list:

Additional arguments:

size: the number of items visible in the list (default is "1")

multiple: if set to "true", any number of items may be selected (default is "false")

10. Hidden Field

`<input type="hidden" name="hiddenField" value="nyah"><-- right there, don't you see it?`

A hidden field: `<-- right there, don't you see it?`

What good is this?

All input fields are sent back to the server, including hidden fields

This is a way to include information that the user doesn't need to see (or that you don't want her to see)

The value of a hidden field can be set programmatically (by JavaScript) before the form is submitted

Example:

`<html>`

`<head>`

`<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">`

`</head>`

`<body>`

`<p>Who are you?</p>`

`<form method="post" action="">`

`<p>Name:<input type="text" name="textfield"> </p>`

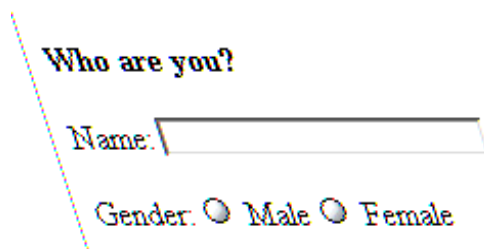
`<p>Gender: <input type="radio" name="gender" value="m">Male <input type="radio" name="gender"`

`value="f">Female</p>`

`</form>`

`</body>`

`</html>`



Dynamic HTML:

DHTML is a new and emerging technology that has evolved to meet the increasing demand for eye-catching and mind-catching web sites. DHTML combines HTML with cascading style sheets (CSSs) and scripting languages. HTML

specifies a web page's element like table, frame, paragraph, butteted list etc. Cascading style sheets can be used to determine an element's size, color , position, and a no of other features. Scripting language can be used to manipulate the web page's elements so that styles assigned to them can change in response to a user's input.

Cascading Style Sheets:

Style sheets are used for adding styles (eg. Fonts, colors, spacing) to web documents. It maintains the standard and uniformity throughout a web site and provides numerous attributes to create dynamic effects. With style sheets, text and image formatting properties can be predefined in a single list. The advantage of a style sheet includes the ability to make global changes to all documents for a single location. For CSS, <style>.....</style> tags are used. Between the<style>...</style> html tags, specific style attributes are listed.

The <style> </style> tags are written within the <head>....</head> tags.

Syntax:

```
<style type = "text/css">
```

```
Tag {attribute: value; attribute: value ....}
```

```
...
```

```
</style>
```

Font attributes:

Font-family: a comma- delimited sequence of font family Names (Arial, Cursive).

Font-style: Normal, italic or oblique.

Font-weight: Normal, bold, bolder or one of the nine values (100,200, 300, 400, 500,600,700, 800, 900)

Font size: XX-small, -small, small, medium, large, X-large, xx-large.

Example:

```
<html>
```

```
    <head>
```

```
        <title>Style sheets </title>
```

```
        <style type = "text/css">
```

```
            H1{font-family: Arial: cursive}
```

```
            P{font-size 12pt; font-style:italic}
```

```
        </style>
```

```
    </head>
```

```
    <body>
```

```
        <h1>purbanchal university </h1>
```

```
        <p> this university was established in 2056. There are there constituent college of pu. And all are  
        located at biratnagar. </p> PU has many affiliated colleges among them is the KHEC.....
```

```
    </body>
```

```
</html>
```

Color & background attributes:

Color: an element text color.

Background-color: Specifies the color in an element's background.

Background_image: sets background image.

Example:

```
<html>
  <head>
    <title>CSS</title>
    <style type = "text/css">
      H1{font-family: Arial, Helvetica; font-size:26pt; backgroundimage: KhEC.jpg;}
      H2{font-family:Arial, Helvetica; background-image:KhEC.jpg}
      p{font-size;12pt; font-style:italic; font-weight: bdd; color:red; background-color:red}
    </style>
  </head>
  <body>
    <h1>welcome to purbachal university </h1><br /><br />
    <h2>Introduction </h2>
    <p> the university was established in .....</P>
  </body>
</html>
```

Text attributes:

Text-decoration: Adds decoration to an element's text .name, underline, overline, line-through, blink

Vertical-align: Determines on element's vertical position. Sub, super, top, text-top, middle, bottom, text-bottom.

Text-transform: Applies a transformation to the text. Capitalize (puts the text into initial caps), uppercase, lowercase, or none.

Text-align: Align text within an element. Right , left, centre.

Text-indent: Indents the first lien of text.

Example:

```
<html>
  <head>
    <title>CSS</title>
    <style type = "text/css">
      H1{ font-family: arial, Helvetica; font-size:26pt;textdecoration: blink; color:red }
      P{font-size: 12pt; font-style: normal; font-weight: bold; color:red }
      H6{font-size:12pt; font-style: italic ; color:red; text-indent:.5in}
    </style>
  <head>
  <body>
    <h1> purbanchal university </h1><br />
    <p>This university has 3 constituent colleges <p>
    <ul>
      <li> PUSET</li>
      <li>Management campus</li>
```

```
        <li>CPAD</li>
    </ul>
    <u>KHEC</u>
    <h6> KHEC is an affiliated college of PU. It has the stream like B.E computer and B.E elex &
    comm.</h6>
</body>
</html>
```

Border Attributes:

Border-style: solid, double, groove, inset, outset.

Border-color: color name

Border-width: Thin, medium

Border-top-width: Thin, medium

Border-bottom-width: Thin, medium

Border-left-width: Thin, medium

Border-right-width: Thin, medium

Border-top: Specifies width, color & style.

Border: Sets all the properties at once.

Margin related attributes

Margin-top: percent, length or auto.

Margin-bottom: percent, length or auto.

Margin-left: Percent, length or auto.

Margin-right: percent, length, or auto.

Margin: percent, length, or auto.

Example:

```
<html>
<head>
....
<style type= "text/css">
Body{margin-top:10%}
H1{font-family:arial, Helvetica font-style: italic}
</style></head>
<body>
<h1> purbanchal university </h1>
```

List attributes:

List-style: Disc, circle, square, decimal, lower-roman, upperroman, lower-alpha, upper-alpha.

Example:

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



```
<style type = "text/CSS">
Body{margin-top:10%}
H1{font-family: arial, Helvetica; color: red}
UL{ list-style-type: lower_roman}
</style>
</head>
<body>
<h1> welcome to PU </h1>
<p> Engineering courses offered by PU are
<ul>
<li>B.E Computer</li>
<li>B.E Elx & comm.. </li>
<li>B.e civil</li>
<li>B.Arch. </li>
</ul>
</body>
</html>
```

Class:

Style sheets support classes or sets of style changes for a document. A call can be defined to change the style in a specific way for any element it is applied to and classes can be used to identify logical sets of style changes that might be different for different html elements. The style changes can be applied directly to each html element or applied directly to each html element or applied to part of a document with tags. If any element is made a member of a class by inserting class = class name into its opening tag, it conforms to that class's specification.

Example:

```
<html>
<head>
<title>class </title>
<style type = "text/css">
P{font-size 12pt; font_wight: bold; margin_lef:10%; margin_right:10%}
.question {color: brown; font-style: italic}
.answer {color: red}
</style>
</head>
<body>
<p class = "question">How many students are in B.E computer ? </p>
<p class = "answer">There are 17 students in B.E computer ? </p>
<p class = "question"> What are the subjects taught in 1st semester? </p>
<p class= "answer">The subjects are</p>
</body>
```

```
        <li>comp. concept</li>
        <li>Maths-I</li>
        <li>C-programming</li>
        <li>Engg:Drawing</li>
        <li>Workshop technology</li>
        <li> Communicative Technique. </li>
    </ol>
</body>
</html>
```

External style Sheets:

External style sheets are composed of standard text, which consists of a series of entries, each composed of a selectors & a decoration. The selector indicates the html elements affected by the properties in the declaration. These external style sheets are saved as a file with extension.css, which can be linked, to a web page via the<link> tag.

Syntax:

```
<link rel = stylesheet href = "style sheet file name">
```

Example:

Code for mystyle.css

```
P{font-size:12pt; font-Wight: bold; }
.question {color: brown; font-style:italic}
.answer {color:cyan}
.big{font-size:14pt; text-decoration:underline; texttransformation: uppercase; color:red}
```

code for index.htm

```
<html>
    <head>
        <title>external style sheets</title>
        <link rel= stylesheet href= "mystyle.css">
    </head>
    <body>
        <p class= "question"> what are the subject in <span class="big"> B.E computer </span>first semester.</p>
        <p class = "answer"> The subjects in first semester of <span class = "big"> B.E computer </span> are: </p>
        <ol>
            <li>
            <li>
            .....
        </ol>
    </body>
</html>
```

Chapter 6: JavaScript Introduction

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the same?

- The answer is NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript do?

1. **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
2. **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page
3. **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
4. **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
5. **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
6. **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
7. **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

The Real Name is ECMAScript

JavaScript's official name is ECMAScript.

ECMAScript is developed and maintained by the ECMA organization. ECMA-262 is the official JavaScript standard.

The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The development of ECMA-262 started in 1996, and the first edition of was adopted by the ECMA General Assembly in June 1997.

The standard was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development of the standard is still in progress.

JavaScript How To

The HTML `<script>` tag is used to insert a JavaScript into an HTML page.

Put a JavaScript into an HTML page

The example below shows how to use JavaScript to write text on a web page:

Example:

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

The example below shows how to add HTML tags to the JavaScript:

Example:

```
<html>
```

```
<script type="text/javascript">
document.write("<h1>Hello World!</h1>");
</script>
</body>
</html>
```

Example Explained

To insert a JavaScript into an HTML page, we use the `<script>` tag. Inside the `<script>` tag we use the `type` attribute to define the scripting language.

So, the `<script type="text/javascript">` and `</script>` tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

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

By entering the `document.write` command between the `<script>` and `</script>` tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! To the page:

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

Note: If we had not entered the `<script>` tag, the browser would have treated the `document.write("Hello World!")` command as pure text, and just write the entire line on the page.

How to Handle Simple Browsers

Browsers that do not support JavaScript, will display JavaScript as page content. To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag should be used to "hide" the JavaScript.

Just add an HTML comment tag `<!--` before the first JavaScript statement, and a `-->` (end of comment) after the last JavaScript statement, like this:

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

The two forward slashes at the end of comment line (//) is the JavaScript comment symbol. This prevents JavaScript from executing the --> tag.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Dolly" to the web page:

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

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

Example:

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and ends with a right curly bracket }. The purpose of a block is to make the sequence of statements execute together. This example will write a heading and two paragraphs to a web page:

Example:

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

JavaScript Comments

Comments can be added to explain the JavaScript, or to make the code more readable.

Single line comments start with `//`.

Example:

```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs:
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Multi-Line Comments

Multi line comments start with `/*` and end with `*/`.

The following example uses a multi-line comment to explain the code:

Example:

```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Using Comments at the End of a Line

In the following example the comment is placed at the end of a code line:

Example:

```
<script type="text/javascript">
document.write("Hello"); // Write "Hello"
document.write(" Dolly!"); // Write " Dolly!"
</script>
```

JavaScript Variables

Variables are "containers" for storing information.

Do You Remember Algebra From School?

Do you remember algebra from school? $x=5$, $y=6$, $z=x+y$

Do you remember that a letter (like x) could be used to hold a value (like 5), and that you could use the information above to calculate the value of z to be 11?

These letters are called **variables**, and variables can be used to hold values ($x=5$) or expressions ($z=x+y$).

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions.

A variable can have a short name, like `x`, or a more descriptive name, like `carname`.

Rules for JavaScript variable names:

- Variable names are case sensitive (`y` and `Y` are two different variables)
- Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.

Example:

```
<html>
  <body>
    <script type="text/javascript">
      var firstname;
      firstname="Hege";
      document.write(firstname);
      document.write("<br />");
      firstname="Tove";
      document.write(firstname);
    </script>
    <p>The script above declares a variable, assigns a value to it, displays the value, changes the value,
    and displays the value again.</p>
  </body>
</html>
```

Output:

Hege
Tove

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var statement**:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

Assigning Values to Undeclared JavaScript Variables

If you assign values to variables that have not yet been declared, the variables will automatically be declared.

These statements:

```
x=5;
```

```
carname="Volvo";
```

have the same effect as:

```
var x=5;
```

```
var carname="Volvo";
```

Redeclaring JavaScript Variables

If you redeclare a JavaScript variable, it will not lose its original value.

```
var x=5;
```

```
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;
```

```
z=y+5;
```

JavaScript Operators

= is used to assign values.

+ is used to add values.

The assignment operator = is used to assign values to JavaScript variables.

The arithmetic operator + is used to add values together.

```
y=5;
```

```
z=2;
```

```
x=y+z;
```

The value of x, after the execution of the statements above is 7.

JavaScript Arithmetic Operators

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

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
```

```
txt2="nice day";
```

```
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
```

```
txt2="nice day";
```

```
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
```

```
txt2="nice day";
```

```
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is: **If you add a number and a string, the result will be a string!**

Example:

```
x=5+5;
```

```
document.write(x);
```

```
x="5"+"5";
```

```
document.write(x);
```

```
x=5+"5";
```

```
document.write(x);
```

```
x="5"+5;
```

```
document.write(x);
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How can it be used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

You will learn more about the use of conditional statements in the next chapter of this tutorial.

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	And	(x < 10 && y > 1) is true
	Or	(x==5 y==5) is false
!	Not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax:

```
variablename=(condition)?value1:value2
```

Example:

```
Greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President" else it will be assigned "Dear".

JavaScript If...Else Statements

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

If-statement is used to execute some code only if a specified condition is true.

Syntax:

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

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example:

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

Notice that there is no else in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

If...else Statement

if....else statement is used to execute some code if a condition is true and another code 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 not true  
}
```

Example:

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

//Otherwise you will get a "Good day" greeting.

```
var d = new Date();
var time = d.getHours();
if (time < 10) {
    document.write("Good morning!");
}
else {
    document.write("Good day!");
}
</script>
```

If...else if...else Statement

if...else if...else statement is used to select one of several blocks of code to be executed.

Syntax:

```
if (condition1) {
    code to be executed if condition1 is true
}
else if (condition2) {
    code to be executed if condition2 is true
}
else {
    code to be executed if condition1 and condition2 are not true
}
```

Example:

```
<script type="text/javascript">
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>Hello World!</b>");
}
</script>
```

Random link

this example demonstrates a link, when you click on the link it will take you to W3Schools.com OR to RefsnesData.no. There is a 50% chance for each of them.

```
<html>
```

```
<body>
<script type="text/javascript">
var r=Math.random();
if (r>0.5){
    document.write("<a href='http://www.w3schools.com'>Learn Web Development!</a>");
}
else{
    document.write("<a href='http://www.refsnesdata.no'>Visit Refsnes Data!</a>");
}
</script>
</body>
</html>
```

JavaScript Switch Statement

Conditional statements are used to perform different actions based on different conditions.

The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax:

```
switch(n){
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example:

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay){
case 5:
    document.write("Finally Friday");
```

case 6:

```
document.write("Super Saturday");
```

```
break;
```

case 0:

```
document.write("Sleepy Sunday");
```

```
break;
```

default:

```
document.write("I'm looking forward to this weekend!");
```

```
}
```

```
</script>
```

JavaScript Popup Boxes

JavaScript has three kinds of popup boxes: Alert box, Confirm box, and Prompt box.

1. Alert Box

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

Syntax:

```
alert("sometext");
```

Example:

```
<html>
```

```
<head>
```

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

```
function show_alert(){
```

```
alert("I am an alert box!");
```

```
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<input type="button" onclick="show_alert()" value="Show alert box" />
```

```
</body>
```

```
</html>
```

2. Confirm Box

A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax:

```
confirm("sometext");
```

Example:

```
<html>
```

```
<head>
```

```
function show_confirm(){
var r=confirm("Press a button");
if (r==true) {
    alert("You pressed OK!");
}
else {
    alert("You pressed Cancel!");
}
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show confirm box" />
</body>
</html>
```

3. Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax:

```
prompt("sometext","defaultvalue");
```

Example:

```
<html>
<head>
<script type="text/javascript">
function show_prompt(){
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="") {
    document.write("Hello " + name + "! How are you today?");
}
}
</script>
</head>
<body>
<input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```

Alert box with line breaks


```
<html>
<head>
<script type="text/javascript">
function disp_alert(){
alert("Hello again! This is how we" + '\n' + "add line breaks to an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="disp_alert()" value="Display alert box" />
</body>
</html>
```

JavaScript Functions

A function will be executed by an event or by a call to the function.

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function. A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file). Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example

Example:

```
<html>
<head>
```

```
function displaymessage(){
    alert("Hello World!");
}
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: `alert("Hello world!!")` in the example above had not been put within a function, it would have been executed as soon as the line was loaded. Now, the script is not executed before a user hits the input button. The function `displaymessage()` will be executed if the input button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example:

```
<html>
<head>
<script type="text/javascript">
function product(a,b){
    return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
</body>
</html>
```

The Lifetime of JavaScript Variables

If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

Function with a parameter

How to pass a variable to a function, and use the variable in the function.

```
<html>
<head>
<script type="text/javascript">
function myfunction(txt){
alert(txt);
}
</script>
</head>
<body>
<form>
<input type="button" onclick="myfunction('Hello')" value="Call function">
</form>
<p>By pressing the button above, a function will be called with "Hello" as a parameter. The function will alert the
parameter.</p>
</body>
</html>
```

Function that returns a value

```
<html>
<head>
<script type="text/javascript">
function myFunction(){
return ("Hello world!");
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(myFunction())
</script>
</body>
</html>
```

JavaScript for Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

For loop is used when you know in advance how many times the script should run.

Syntax:

```
for (var=startvalue;var<=endvalue;var=var+increment){  
    code to be executed  
}
```

Example:

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

Example:

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

Example:

```
<html>  
<body>  
<script type="text/javascript">  
for (i = 1; i <= 6; i++){  
    document.write("<h" + i + ">This is heading " + i);  
    document.write("</h" + i + ">");  
}  
</script>  
</body>  
</html>
```

JavaScript While Loop

Loops execute a block of code a specified number of times, or while a specified condition is true.

The while Loop

While loop loops through a block of code while a specified condition is true.

Syntax:

```
while (var<=endvalue) {  
    code to be executed  
}
```

Note: The <= could be any comparing operator.

Example:

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

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>
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax:

```
do {  
    code to be executed  
}  
while (var<=endvalue);
```

Example:

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example:

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
do {
```

```
document.write("<br />");  
i++;  
}  
while (i<=5);  
</script>  
</body>  
</html>
```

JavaScript Break and Continue Statements

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example:

```
<html>  
<body>  
<script type="text/javascript">  
var i=0;  
for (i=0;i<=10;i++) {  
    if (i==3) {  
        break;  
    }  
    document.write("The number is " + i);  
    document.write("<br />");  
}  
</script>  
</body>  
</html>
```

The continue Statement

The continue statement will break the current loop and continue with the next value.

Example:

```
<html>  
<body>  
<script type="text/javascript">  
var i=0  
for (i=0;i<=10;i++) {  
    if (i==3) {  
        continue;  
    }  
    document.write("The number is " + i);  
    document.write("<br />");  
}  
}
```

</body>

</html>

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax:

```
for (variable in object) {  
    code to be executed  
}
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example:

Use of for...in statement to loop through an array:

Example:

```
<html>  
<body>  
<script type="text/javascript">  
var x;  
var mycars = new Array();  
mycars[0] = "Saab";  
mycars[1] = "Volvo";  
mycars[2] = "BMW";  
for (x in mycars) {  
    document.write(mycars[x] + "<br />");  
}  
</script>  
</body>  
</html>
```

Examples Of Javascript

1. Digital Clock

```
<script>  
function show_clock(){  
    var today = new Date();  
    var h = today.getHours();  
    var m = today.getMinutes();  
    var s = today.getSeconds();  
    h = addzero(h);  
    m = addzero(m);  
    s = addzero(s);  
    hh = (h>12)?(h-12):h;
```

```
document.getElementById('current_clock').innerHTML="System Current Time = "+hh+": "+m+": "+s+" ["+b+"]";

setTimeout("show_clock()",1000);
}
function addzero(a){
    if(a<10){
        a = "0"+a;
    }
    return a;
}
</script>
<body onload="show_clock();">
<span id="current_clock"></span>
</body>
```

2. Inserting Rows Of a Table

```
<html>
<head>
<script type="text/javascript">
function insRow(){
    var x=document.getElementById('myTable').insertRow(0);
    var y=x.insertCell(0);
    var z=x.insertCell(1);
    y.innerHTML="Web";
    z.innerHTML="Programming";
}
</script>
</head>
<body>
<table id="myTable" border="1">
<tr >
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr >
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
<tr >
<td>Row3 cell1</td>
<td>Row3 cell2</td>
```



```
</table>
<br />
<input type="button" onclick="insRow()" value="Insert row">
</body>
</html>
```

3. Deleting Rows of a Table

```
<html>
<head>
<script type="text/javascript">
function deleteRow(r){
    var i=r.parentNode.parentNode.rowIndex;
    document.getElementById('myTable').deleteRow(i);
}
</script>
</head>
<body>
<table id="myTable" border="1">
<tr >
    <td>Row 1</td>
    <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
</tr>
<tr >
    <td>Row 2</td>
    <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
</tr>
<tr >
    <td>Row 3</td>
    <td><input type="button" value="Delete" onclick="deleteRow(this)"></td>
</tr>
</table>
</body>
</html>
```

4. Counting Images in a Web Page

```
<html>
<body>
<br />
<br /><br />
<script type="text/javascript">
document.write("This document contains: " + document.images.length + " images.");
```

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

5. Degree to Celcius Converstion Vice Versa

```
<html>
<head>
<script type="text/javascript">
function convert(degree){
if (degree=="C") {
    F=document.getElementById("c").value * 9 / 5 + 32;
    document.getElementById("f").value=Math.round(F);
}
else {
    C=(document.getElementById("f").value -32) * 5 / 9;
    document.getElementById("c").value=Math.round(C);
}
}
</script>
</head>
<body>
<p></p><b>Insert a number into one of the input fields below:</b></p>
<form>
<input id="c" name="c" onkeyup="convert('C')"> degrees Celsius<br />
equals<br />
<input id="f" name="f" onkeyup="convert('F')"> degrees Fahrenheit
</form>
<p>Note that the <b>Math.round()</b> method is used, so that the result will be returned as an integer.</p>
</body>
</html>
```

6. Changing Image on Mouse Hovering

```
<html>
<head>
<script type="text/javascript">
function mouseOver(){
    document.getElementById("b1").src ="b_blue.gif";
}
function mouseOut(){
    document.getElementById("b1").src ="b_pink.gif";
}
</script>
```

```
<body>
  <a href="http://www.w3schools.com" target="_blank">
    </a>
</body>
</html>
```

7. Changing Background and foreground color

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Foreground and Background Changing</title>
<script type="text/javascript">
  function changebgcolor(abc)
  {
    if(abc == "select")
      alert("Please Select Color !");
    else
      document.fgColor = abc;
  }
  function changebcolor(abc)
  {
    if(abc == "select")
      alert("Please Select Color !");
    else
      document.bgColor = abc;
  }
</script>
</head>
<body>
<h1>Khwopa Engineering College</h1>
<p>This is a test from khwopa Engineering College</p>
<form name="frm1" method="post" action="">
Foreground Color
<select name="fgcolor" onchange="changebgcolor(this.value);">
  <option value="select">Please Select</option>
  <option value="RED">RED</option>
  <option value="GREEN">GREEN</option>
  <option value="BLUE">BLUE</option>
  <option value="PURPLE">PURPLE</option>
  <option value="MAROON">MAROON</option>
```

```
<option value="ORANGE">ORANGE</option>
<option value="WHITE">WHITE</option>
<option value="PINK">PINK</option>
</select>
<hr/>
Background Color
<select name="bcolor" onchange="changebcolor(this.value);">
  <option value="select">Please Select</option>
  <option value="RED">RED</option>
  <option value="GREEN">GREEN</option>
  <option value="BLUE">BLUE</option>
  <option value="PURPLE">PURPLE</option>
  <option value="MAROON">MAROON</option>
  <option value="BLACK">BLACK</option>
  <option value="ORANGE">ORANGE</option>
  <option value="WHITE">WHITE</option>
  <option value="PINK">PINK</option>
</select>
</form>
</body>
</html>
```

8. Capturing the Coordinate of an Image

```
<html>
<head>
  <title>Capturing the coordinate of a mouse pointer</title>
  <script language=javascript>
    function findco()
    {
      var x,y;
      x=event.screenX;
      y=event.screenY;
      document.frm.t1.value=x;
      document.frm.t2.value=y;
    }
  </script>
</head>
<body>
  <form name=frm>
    X Coordinates<input type=text name=t1> <br>
    Y Coordinates<input type=text name=t2> <br> <br>
```

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

CHAPTER: 7. Open Source Programming with PHP

PHP is a powerful tool for making dynamic and interactive Web pages. PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP. In our PHP tutorial you will learn about PHP, and how to execute scripts on your server.

What is PHP?

- PHP stands for **PHP: Hypertext Preprocessor**
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php", ".php3", or ".phtml"

What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is FREE to download from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

PHP Syntax

The PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

A PHP script always starts with **<?php** and ends with **?>**. A PHP script can be placed anywhere in the document. On servers with shorthand-support, you can start a PHP script with **<?** and end with **?>**. For maximum compatibility, we recommend that you use the standard form (**<?php**) rather than the shorthand form.

Different way of writing php code

1. **<?php** echo "Wel Come to PHP Programming"; **?>**
2. **<?PHP** echo "Wel Come to PHP Programming"; **?>**
3. **<?** echo "Wel Come to PHP Programming"; **?>** [with Short open Tag]
4. **<script language="php">** echo "Wel Come To PHP Programming"; **</script>**

A PHP file must have a .php extension. A PHP file normally contains HTML tags, and some PHP scripting code.

<html>

<body>

```
<?php echo "Hello World"; ?>
</body>
</html>
```

Each code line in PHP must end with a **semicolon**. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**.

Comments in PHP

In PHP, we use *//* to make a one-line comment or */** **and** **/* to make a comment block:

```
<html>
<body>
<?php
    //This is a comment
    /*
        This is a comment block
    */
?>
</body>
</html>
```

PHP Variables

As with algebra, PHP variables are used to hold values or expressions. A variable can have a short name, like *x*, or a more descriptive name, like *carName*.

Rules for PHP variable names:

- Variables in PHP starts with a \$ sign, followed by the name of the variable
- The variable name must begin with a letter or the underscore character
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- A variable name should not contain spaces
- Variable names are case sensitive (*y* and *Y* are two different variables)

Creating (Declaring) PHP Variables

PHP has no command for declaring a variable. A variable is created the moment you first assign a value to it:

```
$a="Khwopa";
```

After the execution of the statement above, the variable **\$a** will hold the value **Khwopa**.

Tip: If you want to create a variable without assigning it a value, then you assign it the value of *null*.

Let's create a variable containing a string, and a variable containing a number:

```
<?php
    $txt="Hello World!";
    $x=16;
?>
```

Note: When you assign a text value to a variable, put quotes around the value.

PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it. In the example above, notice that we did not have to tell PHP which data type the variable is. PHP automatically converts the variable to the correct data type, depending on its value. In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

PHP Variable Scope

The scope of a variable is the portion of the script in which the variable can be referenced. PHP has four different variable scopes:

- local
- global
- static

1. Local Scope

A variable declared **within** a PHP function is local and can only be accessed within that function. (the variable has local scope):

```
<?php
$a = 5; // global scope
function myTest(){
    echo $a; // local scope
}
myTest();
?>
```

The script above will not produce any output because the echo statement refers to the local scope variable \$a, which has not been assigned a value within this scope. You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared. Local variables are deleted as soon as the function is completed.

2. Global Scope

Global scope refers to any variable that is defined outside of any function. Global variables can be accessed from any part of the script that is not inside a function. To access a global variable from within a function, use the **global** keyword:

```
<?php
$a = 5;
$b = 10;
function myTest(){
    global $a, $b;
    $b = $a + $b;
}
myTest();
```



```
echo $b;  
?>
```

Output

15.

PHP also stores all global variables in an array called `$GLOBALS[index]`. Its index is the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten as this:

```
<?php  
$a = 5;  
$b = 10;  
function myTest(){  
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];  
}  
myTest();  
echo $b;  
?>
```

3. Static Scope

When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.

To do this, use the **static** keyword when you first declare the variable:

```
Static $rememberMe;
```

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

Parameters

A parameter is a local variable whose value is passed to the function by the calling code. Parameters are declared in a parameter list as part of the function declaration:

```
function myTest($para1,$para2,...){  
    // function code  
}
```

Parameters are also called arguments. We will discuss them in more detail when we talk about functions.

PHP Array Introduction

The array functions allow you to manipulate arrays. PHP supports both simple and multi-dimensional arrays. There are also specific functions for populating arrays from database queries.

Types of An array

1. Indexed Array or Numeric Array
2. Associative Array
3. Multidimensional Array

1. Indexed Array

An array having the index in numeric format is called indexed or numeric array.

Example:

```
$index = array();  
$index[0] = "Ganesh Ram Suwal";  
$index[1] = "30";  
$index[2] = "Male";
```

Or

```
$index = array("Ganesh Ram Suwal", "30", "Male");
```

Print_r function is used to print the content of an array.

Example:

```
<?PHP  
Print_r($index);  
Or  
Echo "<pre>";  
Print_r($index);  
Echo "</pre>";  
?>
```

2. Associative Array

An array having the index in non-numeric format is called indexed or numeric array.

Example:

```
$index = array();  
$index['name'] = "Ganesh Ram Suwal";  
$index['age'] = "30";  
$index['gender'] = "Male";  
Or  
$index = array("name"=>"Ganesh Ram Suwal", "age"=>"30", "gender"=>"Male");
```

Example:

```
<?PHP  
Print_r($index);  
Or  
Echo "<pre>";  
Print_r($index);  
Echo "</pre>";  
?>
```

3. Multidimensional Array

```
$student = array("name"=>array("name1"=>"Ruksar" "name2"=>"smiriti" "name3"=>"Suraj",  
"name4"=>"Sujan"), "gender"=>array("female", "female", "male", "male"));
```

Installation

The array functions are part of the PHP core. There is no installation needed to use these functions.

PHP Array Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
----------	-------------	-----

array()	Creates an array	3
---------	------------------	---

array_change_key_case()	Returns an array with all keys in lowercase or uppercase	4
-------------------------	--	---

array_chunk()	Splits an array into chunks of arrays	4
---------------	---------------------------------------	---

array_combine()	Creates an array by using one array for keys and another for its values	5
-----------------	---	---

array_count_values()	Returns an array with the number of occurrences for each value	4
----------------------	--	---

array_diff()	Compares 4 array values, and returns the differences	4
--------------	--	---

array_diff_assoc()	Compares 4 array keys and values	4
--------------------	----------------------------------	---

	and returns the differences
<code>array_diff_key()</code>	Compares 5 array keys, and returns the differences
<code>array_diff_uassoc()</code>	Compares 5 array keys and values, with an additional user-made function check, and returns the differences
<code>array_diff_ukey()</code>	Compares 5 array keys, with an additional user-made function check, and returns the differences
<code>array_fill()</code>	Fills an 4 array with values
<code>array_filter()</code>	Filters 4 elements of an array using a user-made function
<code>array_flip()</code>	Exchanges 4 all keys with their associated values in an

	array
array_intersect()	Compares 4 array values, and returns the matches
array_intersect_assoc()	Compares 4 array keys and values, and returns the matches
array_intersect_key()	Compares 5 array keys, and returns the matches
array_intersect_uassoc()	Compares 5 array keys and values, with an additional user-made function check, and returns the matches
array_intersect_ukey()	Compares 5 array keys, with an additional user-made function check, and returns the matches
array_key_exists()	Checks if 4 the specified key exists in the array
array_keys()	Returns all 4 the keys of

	an array
array_map()	Sends each 4 value of an array to a user-made function, which returns new values
array_merge()	Merges one 4 or more arrays into one array
array_merge_recursive()	Merges one 4 or more arrays into one array
array_multisort()	Sorts 4 multiple or multi- dimensional arrays
array_pad()	Inserts a 4 specified number of items, with a specified value, to an array
array_pop()	Deletes the 4 last element of an array
array_product()	Calculates 5 the product of the values in an array
array_push()	Inserts one 4 or more elements to the end of

	an array
array_rand()	Returns one 4 or more random keys from an array
array_reduce()	Returns an 4 array as a string, using a user- defined function
array_reverse()	Returns an 4 array in the reverse order
array_search()	Searches 4 an array for a given value and returns the key
array_shift()	Removes 4 the first element from an array, and returns the value of the removed element
array_slice()	Returns 4 selected parts of an array
array_splice()	Removes 4 and replaces specified elements of an array

<code>array_sum()</code>	Returns the sum of the values in an array
<code>array_udiff()</code>	Compares array values in a user-made function and returns an array
<code>array_udiff_assoc()</code>	Compares array keys, and compares array values in a user-made function, and returns an array
<code>array_udiff_uassoc()</code>	Compares array keys and array values in user-made functions, and returns an array
<code>array_uintersect()</code>	Compares array values in a user-made function and returns an array
<code>array_uintersect_assoc()</code>	Compares array keys, and compares array values

	made function, and returns an array
array_uintersect_uassoc()	Compares 5 array keys and array values in user-made functions, and returns an array
array_unique()	Removes 4 duplicate values from an array
array_unshift()	Adds one or 4 more elements to the beginning of an array
array_values()	Returns all 4 the values of an array
array_walk()	Applies a 3 user function to every member of an array
array_walk_recursive()	Applies a 5 user function recursively to every member of an array
arsort()	Sorts an 3 array in reverse

	order and maintain index association
asort()	Sorts an array and maintain index association
compact()	Create array containing variables and their values
count()	Counts elements in an array, or properties in an object
current()	Returns the current element in an array
each()	Returns the current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a

	specified value exists in an array
key()	Fetches a 3 key from an array
krsort()	Sorts an 3 array by key in reverse order
ksort()	Sorts an 3 array by key
list()	Assigns 3 variables as if they were an array
natcasesort()	Sorts an 4 array using a case insensitive "natural order" algorithm
natsort()	Sorts an 4 array using a "natural order" algorithm
next()	Advance the 3 internal array pointer of an array
pos()	Alias of 3 current()
prev()	Rewinds the 3 internal array pointer
range()	Creates an 3

	containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an array in reverse order
shuffle()	Shuffles an array
sizeof()	Alias of count()
sort()	Sorts an array
uasort()	Sorts an array with a user-defined function and maintain index association
uksort()	Sorts an array by keys using a user-defined function
usort()	Sorts an array by values using a user-defined function

PHP Array Constants

PHP: indicates the earliest version of PHP that supports the constant.

Constant	Description	PHP
----------	-------------	-----

	case	
CASE_UPPER	Used with array_change_key_case() to convert array keys to upper case	
	case	
SORT_ASC	Used with array_multisort() to sort in ascending order	
SORT_DESC	Used with array_multisort() to sort in descending order	
SORT_REGULAR	Used to compare items normally	
SORT_NUMERIC	Used to compare items numerically	
SORT_STRING	Used to compare items as strings	
SORT_LOCALE_STRING	Used to compare items as strings, based on the current locale	4

Files:

File is a place where information or data is stored. A file is opened with fopen() as a “stream”, and PHP returns a ‘handle’ to the file that can be used to reference the open file in other functions. Each file is opened in a particular **mode**. A file is closed with fclose() or when your script ends.

File Open Modes

‘r’	Open for reading only. Start at beginning of file.
‘r+’	Open for reading and writing. Start at beginning of file.
‘w’	Open for writing only. Remove all previous content, if file doesn’t exist, create it.
‘a’	Open for writing, but start at END of current content (append).
‘a+’	Open for reading and writing, start at END (append) and create file if it doesn’t exist.

File Open/Close **Example:**

```
<?php
    // open and close file to read
    $storead = fopen('test.txt','r');
    fclose($storead);
    // open and close file to write
    $towrite = fopen('test.txt','w');
    fclose($towrite);
?>
```

Writing Data

To write data to a file use

fwrite(\$file,\$data)

```
<?php
    /* Write to a file
    $file = fopen('khwopa.txt', 'w');           // open file to write
```

```
$text = "This is a test from Khwopa Engineering College";  
fwrite($file, $text);           // write to the file  
echo "File created successfully."  
fclose($file);                 // close file  
?>
```

Reading Data

There are two main functions to read data:

1.fgets(\$handle)

- Reads a line of data at a time

2.fread(\$handle,\$bytes)

- Reads up to \$bytes of data, stops at EOF (end of file)
- Used to read entire file at once

feof(\$handle)

Whether the file has reached the EOF point. Returns true if have reached EOF.

```
$fileName = "khwopa.txt";  
echo "<h3>Method 1: fgets()</h3>";  
$file = fopen($fileName, 'r');           // open file to read  
while (!feof($file)) {                  // read file contents one line at a time  
    echo fgets($file,1024);  
}  
fclose($file);
```

Creating folder

we use mkdir keyword to make folder

Eg mkdir("phpfile",0700)

Here phpfile is a folder name and 0700 is a permission to create folder

Creating file

we use touch keyword to create a file

Eg touch("phpfile/student.txt");

Deleting file

unlink('filename');

unlink("phpfile/student.txt");

Rename (file or directory)

rename('old name', 'new name');

Copy file

copy('source', 'destination');

```
if(file_exists("phpfile"))
{
    $mydir = "phpfile/";
    $d = dir($mydir);
    while($entry = $d->read())
    {
        if($entry != "." && $entry!="..")
        {
            echo $entry."<br>";
            unlink($mydir.$entry);
            echo "=>Deleted<br>";
        }
    }
    $d->close();
    rmdir($mydir);
}
```

Example: Adding two numbers inputted through from using PHP

<?PHP

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

```
{
    $fno = $_POST['fno'];
    $sno = $_POST['sno'];
    $sum = $fno + $sno;
    echo "The Sum of Two Numbers = ".$sum;
}
```

```
}
```

```
?>
```

<html>

<head>

<title>Form value is passed to PHP variable</title>

</head>

<body>

<h1>Adding Two Number inputted from user through Html Form</h1>

<form name="frm1" method="post" action="">

**First Number: <input type="text" name="fno" value="">
Second Number: <input type="text" name="sno" value="">
**

<input type="submit" name="btnAdd" value="ADD">

</form>

</body>

</html>

Chapter 8: Databases Connectivity in PHP

Structured Query Language (SQL) is a very important data manipulation language. It is used in many different types of DBMS such as MySQL, Oracle, DB2, Microsoft SQL Server, and Microsoft Access. In this course, students will learn how to use SQL to retrieve, store or update data in MySQL. The SQL syntax and data types may be different in other

databases but students would manage to learn them very quickly once they have the foundation knowledge gained from this SQL course. Weekly lectures provide practical examples, illustration images, and detailed explanations. Weekly assignments will help students practice and go beyond what they have learned in the class.

1.1 What is MySQL?

MySQL is a true multi-user, multi-threaded SQL database server. SQL (Structured Query Language) is the most popular and standardized database language in the world. MySQL is a client/server implementation that consists of a server daemon mysqld and many different client programs and libraries. SQL is a standardized language that makes it easy to store, update and access information.

For example, you can use SQL to retrieve product information and store customer information for a web site. MySQL is also fast and flexible enough to allow you to store logs and pictures in it.

Version: 3.23.11-alpha Printed: 25 February 2000

General Information about MySQL

The main goals of MySQL are speed, robustness and ease of use. MySQL was originally developed because we needed a SQL server that could handle very large databases an order of magnitude faster than what any database vendor could offer to us on inexpensive hardware. We have now been using MySQL since 1996 in an environment with more than 40 databases containing 10,000 tables, of which more than 500 have more than 7 million rows. This is about 100 gigabytes of mission-critical data. The base upon which MySQL is built is a set of routines that have been used in a highly demanding production environment for many years. Although MySQL is still under development, it already offers a rich and highly useful function set.

- MySQL is the most popular open-source database system.
- MySQL is a database.
- The data in MySQL is stored in database objects called tables.
- A table is a collection of related data entries and it consists of columns and rows.
- Databases are useful when storing information categorically. A company may have a database with the following tables: "Employees", "Products", "Customers" and "Orders".

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

Below is an example of a table called "Persons":

LastName	FirstName	Address	City
Suwal	Ganesh Ram	Kwathandau	Bhaktapur
Chawal	Bikash	Lakolachhe	Bhaktapurs

The table above contains three records (one for each person) and four columns (LastName, FirstName, Address, and City).

Queries

A query is a question or a request.

With MySQL, we can query a database for specific information and have a recordset returned.

Look at the following query:

```
SELECT LastName FROM Persons
```

The query above selects all the data in the "LastName" column from the "Persons" table, and will return a recordset like this:

LastName

Suwal

Chawal

About MySQL Database

One great thing about MySQL is that it can be scaled down to support embedded database applications. Perhaps it is because of this reputation that many people believe that MySQL can only handle small to medium-sized systems.

The truth is that MySQL is the de-facto standard database for web sites that support huge volumes of both data and end users (like Friendster, Yahoo, Google).

The free MySQL database is very often used with PHP.

Create a Connection to a MySQL Database

Before you can access data in a database, you must create a connection to the database. In PHP, this is done with the `mysql_connect()` function.

Syntax: `mysql_connect(servername,username,password);`

Parameter	Description
-----------	-------------

Servername	Optional.
------------	-----------

	Specifies the server to connect to. Default value is "localhost" or "127.0.0.1"
--	--

Username	Optional.
----------	-----------

	Specifies the username to log in with. Default
--	--

value is the
name of the
user that
owns the
server
process

Password Optional.
Specifies
the
password to
log in with.
Default is ""

Note: There are more available parameters, but the ones listed above are the most important.

Example:

In the following example we store the connection in a variable (\$con) for later use in the script. The "die" part will be executed if the connection fails:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
// some code
?>
```

Closing a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the `mysql_close()` function:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
// some code
mysql_close($con);
?>
```

A database holds one or multiple tables.

Create a Database

The CREATE DATABASE statement is used to create a database in MySQL.

Syntax:

CREATE DATABASE *database_name*

To get PHP to execute the statement above we must use the *mysql_query()* function. This function is used to send a query or command to a MySQL connection.

Example:

The following example creates a database called "my_db":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}

if (mysql_query("CREATE DATABASE my_db",$con)) {
    echo "Database created";
}
else {
    echo "Error creating database: " . mysql_error();
}
mysql_close($con);
?>
```

Create a Table

The CREATE TABLE statement is used to create a table in MySQL.

Syntax:

```
CREATE TABLE table_name
(
    column_name1 data_type,
    column_name2 data_type,
    column_name3 data_type,
    ....
)
```

We must add the CREATE TABLE statement to the *mysql_query()* function to execute the command.

Example:

The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

// Create database
```

```
if (mysql_query("CREATE DATABASE my_db", $con)) {  
    echo "Database created";  
}  
else {  
    echo "Error creating database: " . mysql_error();  
}  
  
// Create table  
mysql_select_db("my_db", $con);  
$sql = "CREATE TABLE Persons  
(  
    FirstName varchar(15),  
    LastName varchar(15),  
    Age int  
)";  
  
// Execute query  
mysql_query($sql, $con);  
  
mysql_close($con);  
?>
```

Important: A database must be selected before a table can be created. The database is selected with the `mysql_select_db()` function.

Note: When you create a database field of type varchar, you must specify the maximum length of the field, e.g. `varchar(15)`.

The data type specifies what type of data the column can hold.

Primary Keys and Auto Increment Fields

Each table should have a primary key field. A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the `personID` field as the primary key field. The primary key field is often an ID number, and is often used with the `AUTO_INCREMENT` setting. `AUTO_INCREMENT` automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the `NOT NULL` setting to the field.

Example:

```
$sql = "CREATE TABLE Persons  
(  
    personID int NOT NULL AUTO_INCREMENT,  
    PRIMARY KEY(personID),  
    FirstName varchar(15),
```

LastName varchar(15),

Age int

);

mysql_query(\$sql,\$con);

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

Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

Syntax:

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

INSERT INTO table_name VALUES (value1, value2, value3,...)

The second form specifies both the column names and the values to be inserted:

INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)

To get PHP to execute the statements above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

Example:

In the previous chapter we created a table named "Persons", with three columns; "Firstname", "Lastname" and "Age".

We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
mysql_query("INSERT INTO Persons (Firstname, LastName, Age) VALUES ('Peter', 'Griffin',35)");
mysql_query("INSERT INTO Persons (Firstname, LastName, Age) VALUES ('Glenn', 'Quagmire',33)");
mysql_close($con);
?>
```

Insert Data from a Form into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>
<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
```

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

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php". The "insert.php" file connects to a database, and retrieves the values from the form with the PHP `$_POST` variables. Then, the `mysql_query()` function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES('$_POST[firstname]', '$_POST[lastname]', '$_POST[age]')";
if (!mysql_query($sql,$con)) {
    die('Error: ' . mysql_error());
}
echo "1 record added";
mysql_close($con);
?>
```

The SELECT statement is used to select data from a database.

Select Data from a Database Table

The SELECT statement is used to select data from a database.

Syntax:

SELECT column_name(s) FROM table_name

To get PHP to execute the statement above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

Example:

The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons");
while($row = mysql_fetch_array($result)) {
```

```
echo $row['FirstName'] . " " . $row['LastName'];  
echo "<br />";  
}  
mysql_close($con);  
?>
```

The example above stores the data returned by the `mysql_query()` function in the `$result` variable. We use the `mysql_fetch_array()` function to return the first row from the recordset as an array. Each call to `mysql_fetch_array()` returns the next row in the recordset. The while loop loops through all the records in the recordset. To print the value of each row, we use the PHP `$row` variable (`$row['FirstName']` and `$row['LastName']`).

output:

Ganesh Ram Suwal
Bikash Chawal

Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:

```
<?php  
$con = mysql_connect("localhost","root","");  
if (!$con) {  
    die('Could not connect: ' . mysql_error());  
}  
mysql_select_db("my_db", $con);  
$result = mysql_query("SELECT * FROM Persons");  
echo "<table border='1'>  
<tr >  
<th>Firstname</th>  
<th>Lastname</th>  
</tr>";  
while($row = mysql_fetch_array($result)) {  
    echo "<tr >";  
    echo "<td>" . $row['FirstName'] . "</td>";  
    echo "<td>" . $row['LastName'] . "</td>";  
    echo "</tr>";  
}  
echo "</table>";  
mysql_close($con);  
?>
```

The WHERE clause is used to filter records.

The WHERE clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

To get PHP to execute the statement above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

Example:

The following example selects all rows from the "Persons" table where "LastName='Suwal'":

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons WHERE LastName='Suwal'");
while($row = mysql_fetch_array($result)) {
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br />";
}
?>
```

Output:

Ganesh Ram Suwal

The ORDER BY Keyword

The ORDER BY keyword is used to sort the data in a recordset. The records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

Syntax:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

Example:

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM Persons ORDER BY age");
while($row = mysql_fetch_array($result)) {
    echo $row['FirstName'];
```

```
echo " " . $row['LastName'];  
echo " " . $row['Age'];  
echo "<br />";  
}  
mysql_close($con);  
?>
```

Output:

```
Ganesh Ram Suwal 30  
Bikash Chawal 32
```

Order by Two Columns

It is also possible to order by more than one column. When ordering by more than one column, the second column is only used if the values in the first column are equal:

```
SELECT column_name(s)  
FROM table_name  
ORDER BY column1, column2
```

The UPDATE statement is used to modify data in a table.

Update Data in a Database

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

Syntax:

```
UPDATE table_name  
SET column1=value, column2=value2,...  
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

To get PHP to execute the statement above we must use the *mysql_query()* function. This function is used to send a query or command to a MySQL connection.

Example:

FirstName LastName Age

```
Bikash    Chawal    32  
Ganesh  
Ram       Suwal      30
```

The following example updates some data in the "Persons" table:

```
<?php  
$con = mysql_connect("localhost","root","");  
if (!$con) {  
    die('Could not connect: ' . mysql_error());  
}  
mysql_select_db("my_db", $con);
```

```
mysql_query("UPDATE Persons SET Age=31 WHERE FirstName='Bikash' AND LastName='Chawal'");
mysql_close($con);
?>
```

After the update, the "Persons" table will look like this:

FirstName LastName Age

Bikash	Chawal	31
--------	--------	----

Ganesh	Suwal	30
Ram		

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

Delete Data In a Database

The DELETE FROM statement is used to delete records from a database table.

Syntax:

```
DELETE FROM table_name
WHERE some_column = some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted! To get PHP to execute the statement above we must use the *mysql_query()* function. This function is used to send a query or command to a MySQL connection.

Example:

Look at the following "Persons" table:

FirstName LastName Age

Bikash	Chawal	31
--------	--------	----

Ganesh	Suwal	30
Ram		

The following example deletes all the records in the "Persons" table where LastName='Griffin':

```
<?php
$con = mysql_connect("localhost","root","");
if (!$con) {
    die("Could not connect: ' . mysql_error());
}
mysql_select_db("my_db", $con);
mysql_query("DELETE FROM Persons WHERE LastName='Chawal'");
mysql_close($con);
?>
```

After the deletion, the table will look like this:

FirstName LastName Age

Ganesh	Suwal	30
Ram		

Create an ODBC Connection

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database). With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is how to create an ODBC connection to a MS Access Database:

1. Open the **Administrative Tools** icon in your Control Panel.
2. Double-click on the **Data Sources (ODBC)** icon inside.
3. Choose the **System DSN** tab.
4. Click on **Add** in the System DSN tab.
5. Select the **Microsoft Access Driver**. Click **Finish**.
6. In the next screen, click **Select** to locate the database.
7. Give the database a **Data Source Name (DSN)**.
8. Click **OK**.

Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to set up a DSN for you to use.

Connecting to an ODBC

The `odbc_connect()` function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type. The `odbc_exec()` function is used to execute an SQL statement.

Example:

The following example creates a connection to a DSN called northwind, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('northwind','', '');  
$sql="SELECT * FROM customers";  
$rs=odbc_exec($conn,$sql);
```

Retrieving Records

The `odbc_fetch_row()` function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false. The function takes two parameters: the ODBC result identifier and an optional row number:

```
odbc_fetch_row($rs)
```

Retrieving Fields from a Record

The `odbc_result()` function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name. The code line below returns the value of the first field from the record:

```
$compname=odbc_result($rs,1);
```

The code line below returns the value of a field called "CompanyName":

```
$compname=odbc_result($rs,"CompanyName");
```

Closing an ODBC Connection

The `odbc_close()` function is used to close an ODBC connection.

```
odbc_close($conn);
```

An ODBC Example

The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
<html>
<body>
<?php
$conn=odbc_connect('northwind','');
if (!$conn) {
    exit("Connection Failed: " . $conn);}
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
if (!$rs) {
    exit("Error in SQL");
}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs)) {
    $compname=odbc_result($rs,"CompanyName");
    $conname=odbc_result($rs,"ContactName");
    echo "<tr><td>$compname</td>";
    echo "<td>$conname</td></tr>";
}
odbc_close($conn);
echo "</table>";
?>
</body>
</html>
```

CHAPTER 9: Session & Cookies

Session

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

Note: The `session_start()` function must appear BEFORE the `<html>` tag:

```
<?php session_start(); ?>  
<html>
```

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

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
</html>

<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

Output:

Pageviews=1

In the example below, we create a simple page-views counter. The `isset()` function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();
if(isset($_SESSION['views']))
    $_SESSION['views']=$_SESSION['views']+1;
else
    $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

If you wish to delete some session data, you can use the `unset()` or the `session_destroy()` function.

The `unset()` function is used to free the specified session variable:

```
<?php
session_start();
if(isset($_SESSION['views']))
    unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the `session_destroy()` function:

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

Note: `session_destroy()` will reset your session and you will lose all your stored session data.

Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

The `setcookie()` function is used to set a cookie.

Note: The `setcookie()` function must appear BEFORE the `<html>` tag.

Syntax:

```
setcookie(name, value, expire, path, domain);
```

Example:

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "grsuwal", time()+3600);
?>
```

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

Example:

You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

```
<?php
$expire=time()+60*60*24*30;
setcookie("user", "grsuwal ", $expire);
?>
```

In the example above the expiration time is set to a month ($60 \text{ sec} * 60 \text{ min} * 24 \text{ hours} * 30 \text{ days}$).

How to Retrieve a Cookie Value?

The PHP `$_COOKIE` variable is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
// Print a cookie
echo $_COOKIE["user"];
```


// A way to view all cookies

```
print_r($_COOKIE);
```

```
?>
```

In the following example we use the `isset()` function to find out if a cookie has been set:

```
<html>
```

```
<body>
```

```
<?php
```

```
if (isset($_COOKIE["user"]))
```

```
    echo "Welcome " . $_COOKIE["user"] . "!"<br />";
```

```
else
```

```
    echo "Welcome guest!"<br />";
```

```
?>
```

```
</body>
```

```
</html>
```

How to Delete a Cookie?

When deleting a cookie you should assure that the expiration date is in the past.

Example:

```
<?php
```

```
// set the expiration date to one hour ago
```

```
setcookie("user", "", time()-3600);
```

```
?>
```

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms.

The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>
```

```
<body>
```

```
<form action="welcome.php" method="post">
```

```
Name: <input type="text" name="name" />
```

```
Age: <input type="text" name="age" />
```

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

```
</form>
```

```
</body>
```

```
</html>
```

Retrieve the values in the "welcome.php" file like this:

```
<html>
```

```
<body>
```

```
Welcome <?php echo $_POST["name"]; ?>.<br />
```

```
You are <?php echo $_POST["age"]; ?> years old.
```

</body>

</html>

Project 1:

First of all create the database with *db_name* and create table with able name *tbl_name* with appropriate fields.

Step 1: Create Index Page (index.php)

```
<?PHP
```

```
include_once "db_conn.php";
```

```
include_once "function_class.php";
```

```
include_once "listing.php";
```

```
?>
```

Step2: Create database Connection Page (db_conn.php)

```
<?PHP
```

```
    $conn = mysql_connect("localhost","root","") or die('Error: unable to connect Database'.mysql_errno());
```

```
    if($conn){
```

```
        $db = mysql_select_db('db_student') or die('Error: unable to select Database'.mysql_errno());
```

```
    }
```

```
?>
```

Step3: Create Class Database (function_class.php)

```
<?pHp
```

```
Class function_class{
```

```
public function exec($sql){
    if(empty($sql)){
        return mysql_error();
    }
    else{
        return mysql_query($sql);
    }
}

public function fetch_object($result){
    return mysql_fetch_object($result);
}

public function total_rows($result){
    return mysql_num_rows($result);
}

public function getAll(){
    $sql = "SELECT * FROM `tbl_student`";
    $result = $this->exec($sql);
    return $result;
}

public function selectById($id){
    $sql = "SELECT * FROM `tbl_student` WHERE 1 = 1 AND `id` = '$id'";
    $result = $this->exec($sql);
    $row = $this->fetch_object($result);
    return $row;
}

public function deleteById($id){
    $sql = "DELETE FROM `tbl_student` WHERE 1 = 1 AND `id` = '$id'";
    $this->exec($sql);
}

public function insertRecord($name,$address,$email,$status){
    $sql = "
    INSERT INTO
    `tbl_student`
    SET
    `name` = '$name',
```

```
`address` = '$address',  
`email` = '$email',  
`status` = '$status'  
",  
return $this->exec($sql);  
}
```

```
public function updateRecord($id,$name,$address,$email,$status){  
    $sql = "  
    UPDATE  
    `tbl_student`  
    SET  
    `name` = '$name',  
    `address` = '$address',  
    `email` = '$email',  
    `status` = '$status'  
    WHERE  
        `id` = '$id'  
    ";  
  
    return $this->exec($sql);  
}
```

```
} //end of class  
$funObj= new function_class;  
?>
```

Step4: Listing Page (listing.php)

```
<html>  
<head>  
    <title>Khwopa Engineering College (Content Management System)</title>  
</head>  
<body>  
<h1>Khwopa Engineering College</h1>  
<h3>Student Management System</h3>  
<a href="add_new.php"><input type="button" name="btnAdd" value="Add New"></a>  
<table align="center" width="900" border="1" cellpadding="10" cellspacing="1">  
<?PHP  
$result = $funObj->getAll();  
$count = $funObj->total_rows($result);  
if($count >=1){
```

```
?>
    <tr >
        <th>S.No</th>
        <th>Name</th>
        <th>Email</th>
        <th>Address</th>
        <th>Status</th>
    <th>Action</th>
    </tr>
<?PHP
$sn = 0;
While($row = $funObj->fetch_object($result)){
?>
<tr >
    <td><?PHP echo ++$sn; ?></td>
    <td><?PHP echo ucwords($row->name); ?></td>
    <td><?PHP echo $row->email; ?></td>
    <td><?PHP echo $row->address; ?> </td>
    <td><?PHP echo ($row->status==1)?"Active":"Inactive"; ?> </td>
    <td><a href="view.php?id=<?PHP echo $row->id; ?>">View</a> | <a href="edit_record.php?id=<?PHP echo $row-
    >id; ?>">Edit</a> | <a href="delete.php?id=<?PHP echo $row->id; ?>" onClick="return confirm('Are You sure to delete
    the record ???')">Delete</a></td>
    </tr>
<?PHP
}
}
else{
?>
<tr >
    <td colspan="6">Sorry ! No Record Found !!</td>
    </tr>
<?PHP
}
?>
</table>
</body>
```

Step5: Addition Page (add_new.php)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Add New Record</title>
</head>
<body>
<h1>Khwopa Engineering College</h1>
<h3>Student Management System (Add Student)</h3>
<?PHP
include_once "db_conn.php";
include_once "function_class.php";
if(isset($_POST['btnInsert'])){
    $name = $_POST['name'];
    $email = $_POST['email'];
    $address = $_POST['address'];
    $status = $_POST['status'];
    $query = $funObj->insertRecord($name,$address,$email,$status);
    if($query){
        echo "Record Inserted Successfully";
    }
    else{
        echo "Something Error ! Sorry !!";
    }
}
?>
<a href="index.php">Goto Home Page</a>
<form name="frmAdd" action="" method="post">
<table align="center" width="900" border="1" cellpadding="10" cellspacing="1">
    <tr >
        <td>Name</td>
        <td><input type="text" name="name" id="name" value=""></td>
    </tr>
    <tr >
        <td>Email</td>
        <td><input type="text" name="email" id="email" value=""></td>
    </tr>
    <tr >
        <td>Address</td>
        <td><input type="text" name="address" id="address" value=""></td>
    </tr>
    <tr >
        <td>Status</td>
```

```

        <td>
        <select name="status">
        <option value="1">Active</option>
        <option value="0">Inactive</option>
        </select>
    </td>
</tr>
<tr >
    <td>&nbsp;</td>
    <td><input type="submit" name="btnInsert" id="btnInsert" value="INSERT"></td>
</tr>

</table>
</form>
</body>
</html>

```

Step6: Edit Page (edit_record.php)

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Add New Record</title>
</head>
<body>
<h1>Khwopa Engineering College</h1>
<h3>Student Management System (Edit Student)</h3>
<?PHP
include_once "db_conn.php";
include_once "function_class.php";
if(isset($_POST['btnUpdate'])){
    $name = $_POST['name'];
    $email = $_POST['email'];
    $address = $_POST['address'];
    $status = $_POST['status'];
    $id = $_POST['id'];
    $query = $funObj->updateRecord($id,$name,$address,$email,$status);
    if($query){
        echo "Record Updated Successfully";
    }
}

```

```
        else{
            echo "Something Error ! Sorry !!";
        }
    }
    $id = $_GET['id'];
    $row = $funObj->selectByid($id);
    ?>
    <a href="index.php">Goto Home Page</a>
    <form name="frmAdd" action="" method="post">
    <input type="hidden" name="id" value="<?PHP echo $row->id; ?>">
    <table align="center" width="900" border="1" cellpadding="10" cellspacing="1">
        <tr >
            <td>Name</td>
            <td><input type="text" name="name" id="name" value="<?PHP echo $row->name; ?>"></td>
        </tr>
        <tr >
            <td>Email</td>
            <td><input type="text" name="email" id="email" value="<?PHP echo $row->email; ?>"></td>
        </tr>
        <tr >
            <td>Address</td>
            <td><input type="text" name="address" id="address" value="<?PHP echo $row->address; ?>"></td>
        </tr>
        <tr >
            <td>Status</td>
            <td>
                <select name="status">
                    <option value="1" <?PHP if($row->status==1){ echo "selected"; } ?>>Active</option>
                    <option value="0" <?PHP if($row->status==0){ echo "selected"; } ?>>Inctive</option>
                </select>
            </td>
        </tr>
        <tr >
            <td>&nbsp;</td>
            <td><input type="submit" name="btnUpdate" id="btnUpdate" value="UPDATE"></td>
        </tr>
    </table>
    </form>
    </body>
    </html>
```


Step7: Viewing Page (view.php)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Add New Record</title>
</head>
<body>
<h1>Khwopa Engineering College</h1>
<h3>Student Management System (View Student)</h3>
<?PHP
include_once "db_conn.php";
include_once "function_class.php";
$id = $_GET['id'];
$row = $funObj->selectByid($id);
?>
<a href="index.php">Goto Home Page</a>
<table align="center" width="900" border="1" cellpadding="10" cellspacing="1">
  <tr >
    <td>Name</td>
    <td><?PHP echo $row->name; ?></td>
  </tr>
  <tr >
    <td>Email</td>
    <td><?PHP echo $row->email; ?></td>
  </tr>
  <tr >
    <td>Address</td>
    <td><?PHP echo $row->address; ?></td>
  </tr>
  <tr >
    <td>Status</td>
    <td><?PHP echo ($row->status==1)?"Active":"Inactive"; ?></td>
  </tr>
</table>
</body>
</html>
```

Step9:Delete Page (delete.php)

```
<?PHP
include_once "db_conn.php";
include_once "function_class.php";
$id = $_GET['id'];
$funObj->deleteByid($id);
header('Location:index.php');
?>
```

Er. Ganesh Ram Suwal.

Lecturer

Khwopa Engineering College

URL: www.ganeshramsuwal.com.np

e-Mail : suwalganesh@gmail.com

References:

- Introduction to Web Programming and PHP (Ivan Byross)
- www.w3schools.com
- www.php.net



~ Best of LUCK ~