

CHAPTER - 4

ELEMENTARY SERVER SIDE SCRIPTING USING PHP

Dreamweaver :

Dreamweaver is a website building software. Dreamweaver is considered WYSIWYG (what you see is what you get) meaning that when you format your web page, you see the result of formatting instead of the mark-ups that are used for formatting. It was first developed by Macromedia and then was acquired by Adobe in 2005. The software has gone through multiple iterations and is now a part of the Adobe Creative Cloud suite. This means that you'll need to purchase a monthly subscription in order to use the software.

To put it simply, Dreamweaver is software that lets you build, design, and publish websites. It's a native application when means that you install it on your PC or Mac.

It's a flexible application in that you can build your website entirely through the visual editor, or solely via code. You can also use the two together (provided you know the basics of coding), to see how each affects the other.

It also supports a wide variety of languages required to build any type of website like HTML, HTML5, CSS, PHP, Javascript, and jQuery. You can use and edit files that are in other languages, but the code support features will not be available.

XHTML

XHTML stands for Extensible Hyper Text Markup Language. XHTML was developed by World Wide Web Consortium to help the web developers make the transition from HTML to XML. XHTML documents need to be well formed, the website will be compatible with present and future web browsers and rendered more accurately. It also makes website easier to maintain, convert and format in the long run

XHTML combines strength of HTML and XML; thus XHTML pages can be parsed by any XML enabled devices unlike HTML, which requires a lenient HTML specific parser. XHTML syntax is very similar to HTML syntax and almost all the html elements are valid in XHTML as well. But when writing an XHTML document, you need to pay a bit extra attention to make your HTML document compliant to XHTML. The few important points while writing an XHTML document are as follows:

1. Xhtml document must have a DOCTYPE declaration at the top of the document.
2. All XHTML tag and attribute names must be written in lower case.
3. All the tags must be nested properly.
4. End tags are required for non empty element.
5. The start tag of an empty element must end with />
6. All the attribute value must be quoted.
7. Attribute minimization is forbidden .

Following is the Comparison Table Between HTML vs XHTML

BASIS FOR COMPARISON	HTML	XHTML
Abbreviation	Hypertext Markup Language.	Extensible Hypertext Markup Language.
Case Sensitive	HTML is not case sensitive.	XHTML is case sensitive.
Internet Media Type	For HTML, it is text/html.	For XHTML, it is application/xhtml+xml.
Expressive	HTML is less expressive.	XHTML is more expressive as compared to HTML.
Application	HTML is an application of SGML.	XHTML is an application of XML.
Format	HTML has Document file format.	XHTML has markup language as type format.
Parser	HTML requires lenient HTML specific parser.	It needs to parse with standard XML parser.
Constraint	In HTML, there is no concern well-formed constraints.	In XHTML, it is concerned with well-formed constraints.

What is CSS ?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Use of CSS :

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page.

HTML was created to describe the content of a web page, like:

```
<h1>This is a heading</h1>
```

```
<p>This is a paragraph.</p>
```

When tags like `<font>`, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page.

CSS Saves a Lot of Work

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file. When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet :

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

With an external style sheet, you can change the look of an entire website by changing just one file. Each page must include a reference to the external style sheet file inside the `<link>` element.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

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

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" file looks like:

```
"mystyle.css"
body {
    background-color: lightblue;
}

h1 {
    color: navy;
    margin-left: 20px;
}
```

Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page :

```
<head>
<style>
body {
    background-color: linen;
}
h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```

Inline Styles

An inline style may be used to apply a unique style for a single element. To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Inline styles are defined within the “style” attribute of the relevant element:

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Assume that an **external style sheet** has the following style for the <h1> element :

```
h1 {  
    color: navy;  
}
```

Then, assume that an **internal style sheet** also has the following style for the <h1> element:

```
h1 {  
    color: orange;  
}
```

If the internal style is defined **after** the link to the external style sheet, the <h1> elements will be “orange” :

```
<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
<style>  
h1 {  
    color: orange;  
}  
</style>  
</head>
```

However, if the internal style is defined **before** the link to the external style sheet, the <h1> elements will be “navy”:

```
<head>
<style>
h1 {
    color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Cascading Order

What style will be used when there is more than one style specified for an HTML element ?

All the styles in a page will “cascade” into a new “virtual” style sheet by the following rules, where number one has the highest priority :

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Outline : An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element “stand out”.

```
p {
    margin: 30px;
    background: yellow;
    border: 1px solid black;
    outline: 1px solid red;
    outline-offset: 15px;
}
```

Introduction of PHP:

PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

PHP scripts can only be interpreted on a server that has PHP installed.

The client computers accessing the PHP scripts require a web browser only.

A PHP file contains PHP tags and ends with the extension “.php”.

Advantages of PHP

Advantages of PHP are as follows :

1. **It's fast.** PHP is embedded in HTML code and its response time is short.
2. **Open Source software.** PHP is open source software and is available free of cost.
3. **Easy to use.** PHP contains many special features and functions needed to create dynamic web pages. The PHP language is designed to be included easily in HTML file.
4. It can run on many operating system. PHP runs on a variety of operating system like windows, LINUX, MAC OS and most varieties of UNIX.
5. It's available on almost all web hosts if you are going to publish websites on a web host, you will find PHP installed on almost all web hosts for free.
6. Technical support is widely available. A large base of user provides free support through email discussion lists.
7. It is secure. The user cannot see the PHP code.
8. It is designed to support database. PHP includes functionality designed to interact with specific database.
9. It is customizable. The open source license allows programmers to modify the PHP software, adding or modifying features as needed to fit their own specific environment.

Scripting Language :

A script is a set of programming instructions that is interpreted at runtime. A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments. The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.

Server side scripts are interpreted on the server while client side scripts are interpreted by the client application. PHP is a server side script that is interpreted on the server while JavaScript is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages.

Programming Language Vs Scripting Language

Programming language	Scripting language
● Has all the features needed to develop complete applications.	● Mostly used for routine tasks.
● The code has to be compiled before it can be executed.	● The code is usually executed without compiling.
● Does not need to be embedded into other languages.	● Is usually embedded into other software environments.

Use of PHP :

You have obviously heard of a number of programming languages out there; you may be wondering why we would want to use PHP as our poison for the web programming. Below are some of the compelling reasons.

- PHP is **open source and free**.
- Short learning curve compared to other languages such as JSP, ASP etc.
- Large community document

- Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.
- PHP is regular updated to keep abreast with the latest technology trends.
- Other benefit that you get with PHP is that it's a **server side scripting language**; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
- PHP has **in built support for working hand in hand with MySQL**; this doesn't mean you can't use PHP with other database management systems. You can still use PHP with
 - Postgres
 - Oracle
 - MS SQL Server
 - ODBC etc.
- PHP is **cross platform**; this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the \$ sign, followed by the name of the variable :

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

After the execution of the statements above, the variable **\$txt** will hold the value **Hello world!**, the variable **\$x** will hold the value **5**, and the variable **\$y** will hold the value **10.5**.

Note :

- When you assign a text value to a variable, put quotes around the value.
- Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data.

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables :

- A variable starts with the \$ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case-sensitive (\$age and \$AGE are two different variables).

Remember that PHP variable names are case-sensitive.

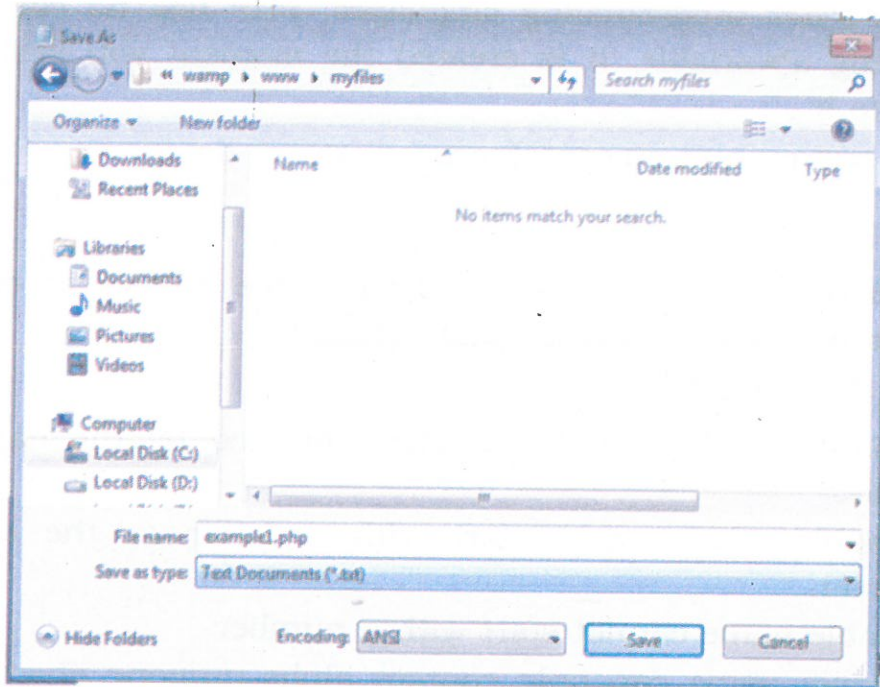
Running PHP Files

The steps in creating and running a PHP file are as follows :

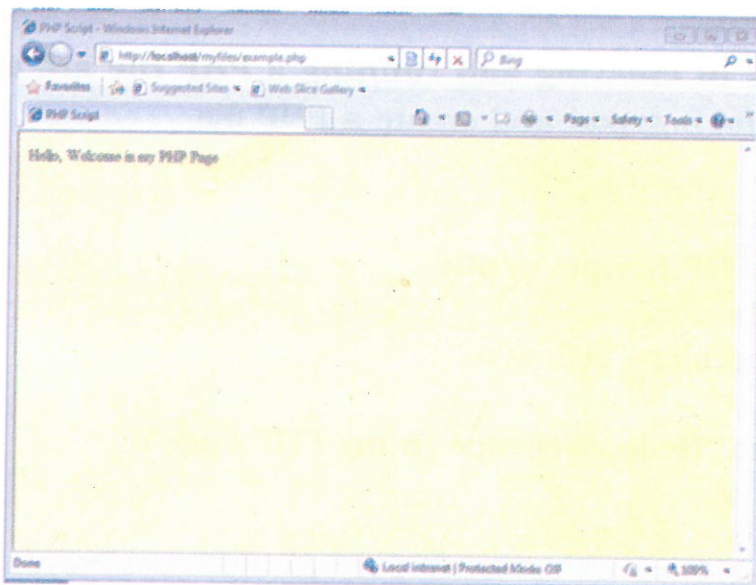
1. Open the notepad and create a PHP file

```
<html>
<head>
<title> PHP Script </title>
</head>
<body bgcolor="yellow">
<?php
    Echo "Hello, Welcome in my PHP Page";
?>
</body>
</html>
```

2. Save it as example1.php. Save this file in directory c:\\wamp\\www\\ or in the directory where you have installed wamp server in root directory as shown



3. Start the wamp server, by running start wamp server application and click on start services.
4. Now click on the localhost which run a default browser. The output is displayed as follows.



Output Variables

The PHP echo statement is often used to output data to the screen. The following example will show how to output text and a variable:

```
<?php
$txt = "GSSS Malyawar";
echo "I love $txt!";
?>
```

The following example will produce the same output as the example above :

```
<?php
$txt = "GSSS Malyawar";
echo "I love " . $txt . "!";
?>
```

The following example will output the sum of two variables:

```
<?php
$x = 15;
$y = 41;
echo $x + $y;
?>
```

PHP is a Loosely Typed Language

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 other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes :

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

```
<?php
$x = 5; // global scope
function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
echo "<p>Variable x outside function is: $x</p>";
?>
```

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

Example

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

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.

PHP The global Keyword

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function) :

Example

```
<?php
$x = 5;
$y = 10;
function myTest() {
    global $x, $y;
    $y = $x + $y;
}
myTest();
echo $y; // outputs 15
?>
```

PHP also stores all global variables in an array called `$GLOBALS[index]`. The *index* holds 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 like this :

```
<?php
$x = 5;
$y = 10;
function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.

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

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

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.

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen. The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup) :

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Display Variables

The following example shows how to output text and variables with the echo statement:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "GSSS Malyawar";
$x = 5;
$y = 4;
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

The PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

Display Variables

The following example shows how to output text and variables with the print statement :

```
<?php
$txt1 = "Learn PHP";
$txt2 = "GSSS Malyawar";
$x = 5;
$y = 4;
print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

The PHP printf() Statement

The printf() statement is useful when we want to output a mix of static text and dynamic information stored within one or several variables. It's ideal for two reasons. First, it separate the static and dynamic data into two distinct sections, allowing for easy maintainance. Second printf() allows you to have considerable control over how the dynamic information is rendered to the screen in term of its type, precision, alignment and position. The syntax is

integer printf(string format[, mixed args])

for example, suppose you want to insert a single dynamic integer value into an other static string.

```
Printf("Neelam Hotel consume : %d bottles of cold drink",1000);
```

The output will be

Neelam Hotel Consume : 1000 bottles of cold drink

In the above example %d is a placeholder known as type specifier and d indicate an integer value will be placed in that position. Commonly used type specifiers are given below :

Type Description

%b	Argument considered an integer; presented as a binary number.
%c	Argument considered an integer; presented as a character corresponding to that ASCII value.
%d	Argument considered an integer; presented as a signed decimal number.
%f	Argument considered a floating point integer; presented as a floating point number.
%o	Argument considered an integer; presented as an octal number.
%s	Argument considered a string; presented as a string.
%u	Argument considered an integer; presented as a unsigned decimal number.
%x	Argument considered an integer; presented as a lowercase hexadecimal number.
%X	Argument considered an integer; presented as a uppercase hexadecimal number..

PHP sprintf() statement

The `sprintf()` statement is functionally identical to `printf()` statement except that the output is assigned to a string rather than rendered to the browser. The syntax is :

String `sprintf( string format [,mixed args])`

For example

```
$cost= sprintf("%.2f",14.2);
```

\$cost store the value 14.20.

Program of add three number

```
<?php
```

```
$a = 234;
```

```
$b = 516;
```

```
$c = 345;
```

```

$s = $a+$b+$c;
Echo "Sum of ".$a ." , ".$b." and ".$c. " = ".$s ;
// also display output with printf statement
printf("Sum of %d, %d and %d = %d",$a, $b, $c, $s) ;
?>

```

Output is : Sum of 234,516 and 345 =1095

Program to find the area and perimeter of a rectangle

```

<?php
$L = 20;
$B = 15;
$a=$L * $B;
$p=2*($L+$B);
Echo "Area of Rectagle=".$a;
Echo "Perimeter of Rectagle=".$p;
?>

```

Program to find the reversed Number of a three digit Number

```

<?php
$n = 756;
$u = $n%10;
$n = $n/10;
$d = $n%10;
$n = $n/10;
$h = $n%10;
$r=$u 100* $d*10+$h;
Echo "Reverse Number is =".$r;
?>

```

Output is : Reverse Number is 657

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types :

- String

- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

- A string is a sequence of characters, like "Hello world!".
- **A string can be any text inside quotes. You can use single or double quotes :**

```
<?php
$x = "Hello world!";
$y = 'Hello world!';
echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers :

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

In the following example \$x is an integer. The PHP var_dump() function returns the data type and value :

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form. In the following example \$x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true; $y = false;
```

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable. In the following example \$cars is an array. The PHP var_dump() function returns the data type and value :

```
<?php
$cars = array("Maruti","BMW","Toyota");
var_dump($cars);
?>
```

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Object

An object is a data type which stores data and information on how to process that data. In PHP, an object must be explicitly declared. First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods :

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}
// create an object
$herbie = new Car();
// show object properties
echo $herbie->model;
?>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL. A variable of data type NULL is a variable that has no value assigned to it.

Variables can also be emptied by setting the value to NULL :

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

PHP String Functions

Look at some commonly used functions to manipulate strings.

Get The Length of a String

The PHP strlen() function returns the length of a string.

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

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

The output of the code above will be: 12.

Count The Number of Words in a String

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

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

The output of the code above will be: 2.

Reverse a String

The PHP strrev() function reverses a string:

```
<?php
echo strrev("Hello world!"); // outputs !dlrowolleH
?>
```

The output of the code above will be: !dlrowolleH.

Search For a Specific Text Within a String

The PHP strpos() function searches for a specific text within a string.

If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

The example below searches for the text "world" in the string "Hello world!" :

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

The output of the code above will be: 6.

Tip: The first character position in a string is 0 (not 1).

Replace Text Within a String

The PHP `str_replace()` function replaces some characters with some other characters in a string.

The example below replaces the text “world” with “Dolly”:

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

The output of the code above will be: Hello Dolly!

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script. A valid constant name starts with a letter or underscore (no \$ sign before the constant name).

Note : Unlike variables, constants are automatically global across the entire script.

Create a PHP Constant

To create a constant, use the `define()` function.

Syntax

`Define(name, value, case-insensitive)`

Parameters :

- name: Specifies the name of the constant
- value: Specifies the value of the constant
- case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

The example below creates a constant with a **case-sensitive** name:

```
<?php
define("GREETING", "Welcome to GSSS Malyawar!");
echo GREETING;
?>
```

The example below creates a constant with a **case-insensitive** name:

```
<?php
define("GREETING", "Welcome to GSSS Malyawar!", true);
echo greeting;
?>
```

Constants are Global

Constants are automatically global and can be used across the entire script. The example below uses a constant inside a function, even if it is defined outside the function:

```
<?php
define("GREETING", "Welcome to GSSS Malyawar!");
function myTest() {
    echo GREETING;
}
myTest();
?>
```

COMMENTS

A comment is a non-executable statement that helps to read and understand a program. Comments may appear anywhere in the program. PHP has two kinds of comments: single line comments and multiline comments.

Single line comments: it starts with double slash symbol (//) and terminates at the end of current line. For example

```
// My first web page in PHP
```

Multiline Comment : the multiline comment begins with /* and ends with */. The multiline comment can be written as

```
/* function for
To add two
number. */
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups :

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

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc. let \$x=5 and \$y=3

Operator	Name	Example	Result	Show it
+	Addition	\$x + \$y	Sum of \$x and \$y	8
-	Subtraction	\$x - \$y	Difference of \$x and \$y	2
*	Multiplication	\$x * \$y	Product of \$x and \$y	15
/	Division	\$x / \$y	Quotient of \$x and \$y	1
%	Modulus	\$x % \$y	Remainder of \$x divided by \$y	2
**	Exponentiation	\$x ** \$y	Result of raising \$x to the \$y'th power	125

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is “=”. It means that the left operand gets set to the value of the assignment expression on the right. Let x = 10 and y = 5

Assignment	Same as...	Description	Show it
$x = y$	$x = y$	The left operand gets set to the value of the expression on the right	$X = 5$
$x += y$	$x = x + y$	Addition	$X = 15$
$x -= y$	$x = x - y$	Subtraction	$X = 5$
$x *= y$	$x = x * y$	Multiplication	$X = 50$
$x /= y$	$x = x / y$	Division	$X = 2$
$x \% = y$	$x = x \% y$	Modulus	$X = 0$

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string) : Let $x=20$ $y=15$

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

PHP Increment / Decrement Operators

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

The PHP decrement operators are used to decrement a variable's value. If $x=10$ then

Operator	Name	Description	Result	Operator
$++\$x$	Pre-increment	Increments $\$x$ by one, then returns $\$x$	$\$x=11$ Return 11	$++\$x$
$\$x++$	Post-increment	Returns $\$x$, then increments $\$x$ by one	$\$x=11$ Return 10	$\$x++$
$--\$x$	Pre-decrement	Decrements $\$x$ by one, then returns $\$x$	$\$x=9$ Return 9	$--\$x$
$\$x--$	Post-decrement	Returns $\$x$, then decrements $\$x$ by one	$\$x=9$ Return 10	$\$x--$

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$\$x$ and $\$y$	True if both $\$x$ and $\$y$ are true
or	Or	$\$x$ or $\$y$	True if either $\$x$ or $\$y$ is true
xor	Xor	$\$x$ xor $\$y$	True if either $\$x$ or $\$y$ is true, but not both
$\&\&$	And	$\$x \&\& \y	True if both $\$x$ and $\$y$ are true
$ $	Or	$\$x \y	True if either $\$x$ or $\$y$ is true
!	Not	! $\$x$	True if $\$x$ is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

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

PHP Conditional Statements

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

In PHP we have the following conditional statements :

- if statement - executes some code if one condition is true

- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif....else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

PHP - The if...else Statement

The if....else statement executes some code if a condition is true and another code if that condition is false.

Syntax

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

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise :

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - The if...elseif....else Statement

The if....elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

The PHP switch Statement

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

Syntax

```
switch (n) {  
    case label1:  
        code to be executed if n=label1;  
        break;  
    case label2:  
        code to be executed if n=label2;  
        break;  
    case label3:  
        code to be executed if n=label3;  
        break;  
    ...  
    default:  
        code to be executed if n is different from all labels;  
}
```

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. The default statement is used if no match is found.

```
<?php  
$favcolor = "red";  
switch ($favcolor) {  
    case "red":  
        echo "Your favorite color is red!";  
        break;  
    case "blue":  
        echo "Your favorite color is blue!";  
        break;  
    case "green":
```

```

        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
    }
?>

```

Program to find the greatest number among three number

```

<?php
$a = 756;
$b = 365;
$c = 898;
if (a>b)
{
    if($a>$c)
        echo "Greatest number is ".$a;
    else
        echo "Greatest number is ".$c;
}
else
{
    if($b>$c)
        echo "Greatest number is ".$b;
    else
        echo "Greatest number is ".$c;
}
?>

```

Program to find a given year is a leap year or not leap Year

```

<?php
$y = 1704;
if($y%4==0 and $y%100!=0 || $y%400==0)
    echo $y." is leap Year";
else
    echo $y." is not a leap Year";
?>

```

Program to find the day fall on a given date of a month . it is given that given month start with Monday.

```
<?php
$d = 14;
$r= $d%7;
Switch($r)
{
case 1:
echo "on date ".$d." of this month is Monday";
break;
case 2:
echo "on date ".$d." of this month is Tuesday";
break;
case 3:
echo "on date ".$d." of this month is Wednesday";
break;
case 4:
echo "on date ".$d." of this month is Thrusday";
break;
case 5:
echo "on date ".$d." of this month is Friday";
break;
case 6:
echo "on date ".$d." of this month is Saturday";
break;
default:
echo "on date ".$d." of this month is Sunday";
}
?>
```

PHP 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 code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements :

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

The PHP while Loop

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

Syntax

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

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

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

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

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

The example below first sets a variable \$x to 1 (\$x = 1). Then, the do while loop will write some output, and then increment the variable \$x with 1. Then the condition is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5:

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

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the \$x variable to 6, then it runs the loop, and then the condition is checked:

```
<?php
$x = 6;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

The PHP for Loop

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

Syntax

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

Parameters :

init counter : Initialize the loop counter value

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

increment counter : Increases the loop counter value

The example below displays the numbers from 0 to 10:

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

Program to print all the Even Number from 10 to 50

```
<html>
<head>
<title> PHP Script </title>
</head>
<body bgcolor="yellow">
<?php
$i = 10;
while($i<=50)
{
    if($i%2==0)
    echo "<br>Even Number=" . $i;
    $i=$i+1;
}
?>
</body>
</html>
```

Program to find the factorial of a given Number

```
<html>
<head>
<title> PHP Script </title>
</head>
<body bgcolor="yellow">
<?php
$n = 5;
$f=1;
For ($i=$n;$i>0;$i- -)
{
$f=$f*$i;
}
echo "<br>Factorial of the Given Number=".$f;
?>
</body>
</html>
```

Program to find the m raise to power n

```
<html>
<head>
<title> PHP Script </title>
</head>
<body bgcolor="yellow">
<?php
$m = 4;
$n=3;
$t=1;
For ($i=1;$i<=$n;$i++)
{
$t=$t*$m;
}
echo $m."Raise to power ". $n ."=".$t;
?>
</body>
</html>
```

The PHP foreach Loop

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

Syntax

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

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

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

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

Functions

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, we can create our own functions. A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page loads. A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

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

```
familyMember("Borge");
```

```
?>
```

The following example has a function with two arguments (\$fname and \$year):

```
<?php
```

```
function familyMember($fname, $year) {
```

```
    echo "$fnameRefsnes. Born in $year <br>";
```

```
}
```

```
familyMember("Hege", "1975");
```

```
familyMember("Stale", "1978");
```

```
familyMember("Kai Jim", "1983");
```

```
?>
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

```
<?php
```

```
function setHeight($minheight = 50) {
```

```
    echo "The height is : $minheight<br>";
```

```
}
```

```
setHeight(250);
```

```
setHeight(); // will use the default value of 50
```

```
setHeight(155);
```

```
setHeight(80);
```

```
?>
```

PHP Functions - Returning values

To let a function return a value, use the return statement:

```
<?php
```

```
function sum($x, $y) {
```

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

```
    return $z;
```

```
}
```

Note : A function name can start with a letter or underscore (not a number).

Tip : Give the function a name that reflects what the function does!
Function names are NOT case-sensitive.

In the example below, we create a function named “writeMsg()”. The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs “Hello world!”. To call the function, just write its name followed by brackets []:

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg(); // call the function
?>
```

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument (\$fname). When the familyMember() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

```
<?php
function familyMember($fname) {
    echo "$fnameRefsnes.<br>";
}
familyMember("Jani");
familyMember("Hege");
familyMember("Stale");
familyMember("Kai Jim");
```

```
echo "50 + 100 = " . sum(50, 100) . "<br>";  
echo "17 + 43 = " . sum(17, 43) . "<br>";  
echo "12 + 14 = " . sum(12, 14);  
?>
```

Arrays

An array stores multiple values in one single variable:

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

What is an Array ?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Maruti";  
$cars2 = "BMW";  
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

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

```
array();
```

In PHP, there are three types of arrays :

- Indexed arrays - Arrays with a numeric index

- Associative arrays - Arrays with named keys
- Multidimensional arrays - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays :

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

```
$cars = array("Maruti", "BMW", "Toyota");
or the index can be assigned manually:
$cars[0] = "Maruti";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

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

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

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

```
<?php
$cars = array("Maruti", "BMW", "Toyota");
echo count($cars);
?>
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a forloop, like this :

```
<?php
$cars = array("Maruti", "BMW", "Toyota");
```

```

$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>

```

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

```
$age = array("Rajesh"=>"35", "Abhishek"=>"19", "Anjali"=>"17");
```

or:

```
$age['Rajesh'] = "35";
```

```
$age['Abhishek'] = "19";
```

```
$age['Anjali'] = "17";
```

The named keys can then be used in a script:

```

<?php
$age = array("Rajesh"=>"35", "Abhishek"=>"19", "Anjali"=>"17");
echo " Anjali is " . $age['Anjali'] . " years old.";

```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

```

<?php
$age = array("Rajesh"=>"35", "Abhishek"=>"19", "Anjali"=>"17");
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>

```

Form Handling

The PHP superglobals `$_GET` and `$_POST` are used to collect form-data.

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

Example

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

The output could be something like this:

Welcome John

Your email address is john.doe@example.com

The same result could also be achieved using the HTTP GET method :

```
<html>
<body>
<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>
Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>
</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

When to use GET ?

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

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

When to use POST ?

Information sent from a form with the POST method is invisible to

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

Developers prefer POST for sending form data.

PHP Form Validation

These pages will show how to process PHP forms with security in mind. Proper validation of form data is important to protect your form from hackers and spammers! The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

PHP Form Validation Example

* required field

Name: *
E-mail: *
Website:

Comment:

Gender: ☐ Female ☐ Male ☐ Other *

Your Input:

The validation rules for the form above are as follows :

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form :

Session Variables

When you work 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 or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. So, Session variables hold information about one single user, and are available to all pages in one application. If you need a permanent storage, you may want to store the data in a database.

A session is started with the `session_start()` function.

Session variables are set with the PHP global variable: `$_SESSION`.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables :

```
<?php
// Start the session
session_start();
```

```

?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>

```

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php"). Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global \$_SESSION variable :

```

<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ". ";
?>
</body>
</html>

```

Another way to show all the session variable values for a user session is to run the following code :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
print_r($_SESSION);
?>
</body>
</html>
```

To change a session variable, just overwrite it. To remove all global session variables or destroy session use `session_unset()` and `session_destroy()` :

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
```

SUMMARY

- Adobe Dreamweaver is a software application that allow you to create and edit HTML and XHTML documents (web pages).
- CSS stand for cascading style sheet. CSS is language for describing the presentation of web pages, including colors,layout and fonts
- WAMP stands for Windows, Apache, MySQL and PHP
- WAMP Server contain PHP admin that allows you to change or add users and for making new databases.
- A PHP Script can be placed anywhere in the document. A PHP script start with <?php and end with ?>
- The default file extension for PHP file is “.php”.
- A comment is a non executable statement that help to read and understand a program
- The variable is a name associated with value. A variable stores or contains the value.
- The scope of a variable is the region of the program in which it defined. A global variable has the global scope; it is defined everywhere in PHP code
- An operator is a symbol that operates on one or more expressions, producing a value that can be assigned to a variable.
- Function is a block of code that is not immediately executed but can be called by your scripts when needed.
- An array is a collection of identical data objects which are stored in consecutive memory location under a common heading or variable name.

TRUE/FALSE

Q.1 State whether the following statements are True or False :

- a) A variable name cannot start with a number
- b) The PHP echo statement is often used to input data to the computer.
- c) An integer can be either positive or negative

- d) An array stores multiple values in one single variable.
- e) constants are automatically global across the entire script.
- f) comparison operators have only true or false Value.
- g) switch statement to select one or many blocks of code to be executed.
- h) Array can store data at consecutive locations.
- i) Do while loop do not execute the statement, while condition is false
- j) A function will execute immediately when a page loads.

Sol. : a) True b) False c) True d) True e) True
 f) True g) False h) True i) False j) False

EXERCISE

- Q.1.** What is the use of Dreamweaver in web designing ?
- Q.2.** What is the use of CSS in web designing ?
- Q.3.** PHP is a server side scripting language ? Explain
- Q.4.** Why conditional statements are used in PHP ? Explain
- Q.5.** PHP is used for designing dynamic websites ? Explain.
- Q.6.** Why session variables are being used ? Explain
- Q.7.** Why functions are being used in PHP ? Explain.
- Q.8.** What is the difference between get and post methods ? Explain.
- Q.9.** What is the difference between while and do while loop in PHP ?
- Q.10.** Explain.
- Q.11.** How can you write a script in PHP ? Explain.
 What do you mean by a form ? How can you design a form?
 Explain.

—End—