CHAPTER 5

INTRODUCTION TO PHP

Chapter Outlines

5.1 INSTALLATION OF PHP

The Windows PHP installer for later versions of PHP is built using MSI technology using the Wix Toolkit (http://wix.sourceforge.net/). It will install and configure PHP and all the built-in and PECL extensions, as well as configure many of the popular web servers such as IIS, Apache, Wamp and Xitami.

First, install your selected HTTP (web) server on your system, and make sure that it works. Then proceed with one of the following install types.

Normal Install : Run the MSI installer and follow the instructions provided by the installation wizard. You will be prompted to select the Web Server you wish to configure first, along with any configuration details needed.

You will then be prompted to select which features and extensions you wish to install and enable. By selecting **"Will be installed on local hard drive"** in the drop-down menu for each item you can trigger whether to install the feature (or) not. By selecting ***"Entire feature will be installed on local hard drive",*** you will be able to install all sub-features of the included feature (for example by selecting this options for the feature "PDO" you will install all PDO Drivers).

The installer then sets up PHP to be used in Windows and the *php.ini* file, and configures certain web servers to use PHP. The installer will currently configure IIS, Apache, Xitami and Sambar Server ; if you are using a different web server you'll need to configure it manually.

Basics of 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. PHP code is executed on the server, and the plain HTML result is sent to the browser.

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. Below, we have an example of a simple PHP script which sends the text **"Hello World"** to the browser :

```
<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.

5.2 PHP BASICS AND FEATURES

Introduction : PHP stands for **"hypertext preprocessor"**, which gives you a good idea of its core purpose. It is a free software that builds dynamic, interactive web sites. PHP is a reflective language. As a program, it can observe and modify itself at the same time that it is being executed.

As a general rule, PHP program run on a web server, and serve web pages to visitors on request, and its complete source code is available for free, so users can customize it to their needs. Its focus is server-side scripting and it can be compared to other languages like ASP.NET by Microsoft and JavaServer Pages (JSPs) by Sun Microsystems in its ability to provide users with dynamic content through web servers. In HTML, PHP is embedded between tags. These tags make it possible to jump between the HTML and PHP.

To make PHP work as server-side scripting, three components are needed: a PHP parser, a programming routine that analyzes and breaks down programming language/sentences into more easily processed functions. PHP parsers are usually CGIs (or) server modules. CGIs are the most common way for users to interact with web servers. Server modules are programming modules that provide services to other programs. The other two required components are a web browser and a web server. PHP creates dynamic web content and can send (or) receive cookies. Cookies are text files sent between the web browser and web server used to analyze user behavior and create customized web content.

5.3 EMBEDDING PHP IN HTML

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. PHP code is executed on the server, and the plain HTML result is sent to the browser.

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document. A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code. Below, we have an example of a simple PHP script which sends the text **"Hello World"** to the browser :

```
<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.

5.4 VARIOUS DATA TYPES WITH EXAMPLES

Data Types : In PHP, a variable does not need to be declared before adding a value to it .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.

Examples :

1. **$name = 'Robbin Jackman';**

```
$salary = 5000;
```

Here **$name** is a string and **$salary** is a number. You can see that these two are two different data types. In PHP we call them **String** and **Integer** respectively.

2. **$name = 'Mr '.$name;**

```
$salary = $salary+200;
```

For **$name** we could do String_Concatenation using dot operator because **$name** is a string. We could use Addition operator on **$salary** because it's an integer.

5.5 DECLARE VARIABLES AND CONSTANTS

Variables in PHP : Variables are used for storing values, like text strings, numbers (or) arrays. When a variable is declared, it can be used over and over again in your script. All variables in PHP start with a $ sign symbol

Syntax :

```
$var_name = value;
```

Example :

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

Naming Rules for Variables :

- A variable name must start with a letter (or) an underscore "_"
- variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_string) (or) with capitalization ($myString).

5.6 VARIOUS OPERATORS

Operators : Operators are used to operate on values. The following are different categories of operators in PHP.

ARITHMETIC OPERATORS :

Operator	Description	Example	Result
+	Addition	x = 2 x + 2	4
–	Subtraction	x = 25 – x	3
*	Multiplication	x = 4 x *5	20
/	Division	15/55/2	32.5
%	Modulus (division remainder)	5% 210% 810% 2	120
++	Increment	x = 5 x + +	x = 6
—	Decrement	x = 5 x —	x = 4

ASSIGNMENT OPERATORS :

Operator	Example	Is The Same As
=	x = y	x = y
+ =	x + = y	x = x + y
– =	x – = y	x = x – y
*=	x * = y	x = x * y
/=	x / = y	x = x/y
.=	x . = y	x = x . y
% =	x % = y	x = x % y

COMPARISON OPERATORS :

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than (or) equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

LOGICAL OPERATORS :

Operator	Description	Example
&&	and	x = 6 y = 3 (x < 10 && y > 1) returns true
||	or	x = 6 y = 3 (x==5 || y==5) returns false
!	not	x = 6 y = 3 !(x==y) returns true

5.7 VARIOUS CONDITIONAL STATEMENTS WITH EXAMPLES

Control Structure : There are two control structures namely conditional statements and looping statements.

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

The following are the conditional statements :

- if statement.
- if...else statement.
- if...elseif....else statement.
- switch statement.

if Statement : Use this statement to execute some code only if a specified condition is true.

Syntax :

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

Example :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!";
?>
</body>
</html>
```

if...else Statement : Use this statement to execute some code if a condition is true and to execute another code if the condition is false.

Syntax :

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

Example :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend!";
else
    echo "Have a nice day!";
?>
</body>
</html>
```

if...elseif....else Statement : Use this statement to select one of several blocks of code to be executed

Syntax :

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

Example :

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

Switch Statement : Use this 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;
```

default :

```
code to be executed if n is different from both label1 and
label2;
}
```

Example :

```
<?php
$destination = "Tokyo";
echo "Traveling to $destination<br />";
switch ($destination)
{
    case "Las Vegas":
            echo "Bring an extra $500";
            break;
    case "Amsterdam":
            echo "Bring an open mind";
            break;
    case "Egypt":
            echo "Bring 15 bottles of SPF 50 Sunscreen";
            break;
    case "Tokyo":
            echo "Bring lots of money";
            break;
    case "Caribbean Islands":
            echo "Bring a swimsuit";
            break;
    default:
            echo "invalid destination";
}
?>
```

5.8 VARIOUS LOOP STATEMENTS WITH EXAMPLES

The Following Looping Statements :

- while loop
- do...while loop
- for loop
- foreach loop

The while Loop : The while loop executes a block of code repeatedly while a condition is true.

Syntax :

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

Example : The example below defines a loop that starts with i=1. 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 :

```
<html>
<body>
<?php
$i=1;
while($i<=5)
 {
 echo "The number is" . $i . "<br />";
 $i++;
 }
?>
</body>
</html>
```

Output :

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

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

Syntax :

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

Example : The example below defines a loop that starts with i=1. It will then increment i with 1, and write some output. Then the condition is checked, and the loop will continue to run as long as i is less than (or) equal to 5:

```
<html>
<body>

<?php
$i=1;
do
 {
 $i++;
 echo "The number is" . $i . "<br />";
 }
while ($i<=5);
?>

</body>
</html>
```

Output :

```
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

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

Syntax :

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

Parameters :

- ***init*** **:** Mostly used to initialize the counter variable.
- ***condition*** **:** Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- ***increment*** **:** Mostly used to increment the counter variable.

Example :

```
<?php
for ($i=1; $i<=5; $i++)
 {
 echo "The number is" . $i . "<br />";
}
?>
```

Output :

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

foreach Loop : The *foreach* loop is used to loop through arrays.

Syntax :

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

Example :

```
<?php
$x=array("one","two","three");
foreach ($x as $value)
 {
 echo $value . "<br />";
 }
?>
```

Output :

```
One
Two
Three
```

5.9 STRINGS AND STRING METHODS

String : It can be defined as a group characters which is terminated by a NULL.

Each character of a string occupies one byte and last character of a string is always the character '\0'.

\0 → Null character and it stands for a character with a value of zero.

The string functions allow you to manipulate strings. Following are the most commonly used string-handling functions which are contained in <string.h>

Function	Description
str_repeat()	Repeats a string a specified number of times.
strlen()	Find the length of a string.
strcmp()	Compares two strings (case-sensitive).
strrev()	Reverses a string.
chr()	Returns a character from a specified ASCII value.

5.10 IMPLEMENT ARRAYS

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

Using an array literal is the easiest way to create a JavaScript Array.

5.10.1 SINGLE AND MULTI DIMENSIONAL ARRAYS

Using an array literal is the easiest way to create a JavaScript Array.

Syntax :

```
var array-name = [item1, item2, ...];
```

Example :

```
var cars = ["Saab", "Volvo", "BMW"];
```

5.10.2 DECLARE AN ARRAY

//Creates an array with no numbered entries.

```
var arr = new Array(5);
test(arr);
// length: 5
var arr = [];
arr.length = 5;
test(arr);
// length: 5
var arr = ['a', 'b', 'c', 'd', 'e'];
```

```
test(arr);
// length:5, 0:a, 1:b, 2:c, 3:d, 4:e
var arr = [undefined, undefined, undefined, undefined,
undefined];
test(arr);
// length:5, 0:undefined, 1:undefined, 2:undefined,
3:undefined, 4:undefined
```

5.10.3 MANIPULATE AN ARRAY

Changing the Length

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f'];
test(arr);
// length:6, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
arr.length = 5;
test(arr);
// length:5, 0:a, 1:b, 2:c, 3:d, 4:e
var arr = ['a','b','c','d','e','f',,,];
test(arr);
// length:8, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
arr.length = 7;
test(arr);
// length:7, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f
```

Removing Entries : JavaScript provides three methods pop, shift and splice that can remove entries from the array and which therefore reduce the length of the array. In each case the value (or values) removed are returned by the call.

// pop() removes the last element from an array.

```
var arr = ['a','b','c','d','e','f'];
var el = arr.pop();
test(arr); // length:5, 0:a, 1:b, 2:c, 3:d, 4:e
console.log(el); // f
```

```
// shift() removes the first element from an array
var arr = ['a','b','c','d','e','f'];
var el = arr.shift();
test(arr); // length:5, 0:b, 1:c, 2:d, 3:e, 4:f
console.log(el); // a
// splice() can remove existing elements
var arr1 = ['a','b','c','d','e','f'];
var arr2 = arr1.splice(0,2); // remove 2 elements starting at index 0
test(arr1); // length:4, 0:c, 1:d, 2:e, 3:f
test(arr2); // length:2, 0:a, 1:b
var arr1 = ['a','b','c','d','e','f',,,'i'];
var arr2 = arr1.splice(6,2); // remove 2 elements starting at index 6
test(arr1); // length:7, 0:a, 1:b, 2:c, 3:d, 4:e, 5:f, 6:i
test(arr2); // length:2
```

5.10.4 ARRAY METHODS

Adding Entries : JavaScript provides e three methods push, unshift and slice for inserting new entries.

```
// push() adds one or more elements to the end of an array
var arr = ['a','b','c','d','e','f',,,,];
arr.push('j');
test(arr);
// length:10, 0:a, 1:b, 2:c, 3:d, 5:f, 9:j
// unshift() adds one or more elements to the beginning of an array
var arr = ['a','b','c','d','e','f',,,,];
arr.unshift('x');
test(arr);
// length:10, 0:x, 1:a, 2:b, 3:c, 4:d, 5:e, 6:f
```

```
arr1 = ['a','b','c','d','e','f',,,'i'];
arr2 = arr1.splice(6,0,'g','h'); // removes 0 elements from index
6, and inserts 'g', 'h'
test(arr1); // length:11, 0:a, 1:b, 2:c, 3:d, 5:f, 6:g, 7:h, 10:i
test(arr2); // length:0
<html>
<body>
<p>The push method appends a new element to an array.
</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
function myFunction() {
   fruits.push("Lemon")
   document.getElementById("demo").innerHTML = fruits;
}
</script>
</body>
</html>
```

5.10.5 PROGRAMS USING ARRAYS AND ARRAY METHODS

Program below illustrates one dimensional array in javascript

```
<script>
var people = ["Joseph","Maria", "Brian", "Susan"];
 document.write(people[0]);
</script>
```

Programs below illustrate multi-dimensional arrays in javascript

```
<script>
var people = [
["joseph", 27, "united states"],
["Maria", 21, "united states" ],
 ["Brian", 35, "united states" ],
 ["Susan", 42, "united states"]
];
 document.write(people[0][0]);//display name joseph
</script>
<script>
var people = [
["joseph", 27, "United states"],
["Maria", 21, "Uraguay" ],
 ["Brian", 35, "United kingdom" ],
 ["Susan", 42, "Australia"]
];
 document.write(people[0][0]);//display name joseph
</script>
<script>
var people = [
["joseph", 27, "United states",["blue","black"]],
["Maria", 21, "Uraguay", ",["brown","green"] ],
 ["Brian", 35, "United kingdom", ",["green","red"] ],
 ["Susan", 42, "Australia", ",["blue","blonde"]]
];
 document.write(people[0][3][0]);//display name joseph eye
colour blue
</script>
<script>
var people = [
["joseph", 27, "United states",["blue","black"]],
```

```
 ["Maria", 21, "Uraguay", ",["brown","green"] ],
 ["Brian", 35, "United kingdom", ",["green","red"] ],
 ["Susan", 42, "Australia", ",["blue","blonde"]]
];
```

People[2][3][1]="brown";

document.write(people[2][3][1]);//display brian hair color as brown

```
</script>
<html>
<body>
<script>
var people = [
["joseph", 27, "United states",["blue","black"]],
["Maria", 21, "Uraguay", ",["brown","green"] ],
 ["Brian", 35, "United kingdom", ",["green","red"] ],
 ["Susan", 42, "Australia", ",["blue","blonde"]]
];
//People[2][3][1]="brown";
 //document.write(people[2][3][1]);//display brian hair color as
brown
for(var i=0;i<people.length;i++)
 {
   document.write("<h2>Person "+i+1+"</h2>);
    for(var details in people[i])
      {
         document.write(people[i][details]+"<br>");
      }
}
</script>
</body>
</html>
```

CHAPTER

6 ADVANCED PHP

Chapter Outlines

6.1 IMPLEMENT FUNCTIONS

6.1.1 DEFINE USER DEFINED FUNCTION

A function in PHP can be built-in (or) user-defined.

User-Defined Functions : The general way of defining a function is,

```
function function_name (arg1, arg2, arg3, ...)
{
statement list
}
Example:-
<html>
<body>
<?php
function writeName()
{
echo "polytechnic college(YSRR)";
}
echo "My college is" ;
writeName();
?>
</body></html>
```

6.1.2 IMPORTANCE OF USER DEFINED FUNCTION

User-defined functions in PHP are essential for several reasons :

1. **Reusability :** User-defined functions in PHP enable code reuse, reducing repetition and promoting efficiency. Organization: Functions help break down code into smaller, focused units, improving readability and maintainability.
2. **Modularity :** Functions promote modular programming by encapsulating specific tasks, enhancing code organization and reusability.

3. **Abstraction :** Functions abstract complex operations, providing a simplified interface for other parts of the code to interact with.

4. **Code Readability :** Well-defined functions make code easier to understand, as each function represents a specific task (or) functionality.

6.1.3 PROCESS OF PASSING ARGUMENTS

Function Arguments and Return Values : Parameters are specified after the function name, inside the parentheses. The following program illustrates the function arguments and its return value.

```
<html>
<body>
<?php
function writeName($fname)
{
echo $fname . "Reddy.<br />";
}
echo "H.O.D(computer dept) name is" ;
writeName("R.Viswanatha");
echo "senior lecturer name is" ;
writeName("N.V.Ramana");
echo "My name is ";
writeName("R.Veera");
?>
</body>
</html>
```

Output :

```
H.O.D (Computer Dept.) name is R.Viswanatha Reddy
Senior lecturer name is N.V.Ramana Reddy
My name is R.Veera Reddy
```

6.1.4 SCOPE AND LIFETIME OF VARIABLES

Every function has its own set of variables. Any variables used outside the function's definition are not accessible. When a function starts, its function parameters are defined. When you use new variables inside a function, they are defined within the function only and don't hang around after the function call ends. In the following example, the variable $ var is not changed by the function call :

EXAMPLE-1

```
function func ()
{
$ var = 2;
}
$var = 1;
func();
print $var;
```

When the function func is called, the variable $var, which is assigned 2,

is only in the scope of the function and thus does not change $var outside the function. The code snippet prints out 1.

EXAMPLE-2

```
function func ()
{
$GLOBALS["var"] = 2;
}
$var = 1;
func();
print $var;
```

It prints the value 2. A global keyword also enables you to declare what global variables you want to access, causing them to be imported into the function's scope.

Function Arguments and Return Values : Parameters are specified after the function name, inside the parentheses. The following program illustrates the function arguments and its return value.

```
<html>
<body>
<?php
function writeName($fname)
{
echo $fname . "Reddy.<br />";
}
echo "H.O.D(computer dept) name is" ;
writeName("R.Viswanatha");
echo "senior lecturer name is" ;
writeName("N.V.Ramana");
echo "My name is ";
writeName("R.Veera");
?>
</body>
</html>
```

Output :

```
H.O.D (Computer Dept.) name is R.Viswanatha Reddy
Senior lecturer name is N.V.Ramana Reddy
My name is R.Veera Reddy
```

INTERNAL FUNCTIONS (BUILT-IN FUNCTIONS) :

Array Functions :

Function	Description
array()	Creates an array.
array_product()	Calculates the product of the values in an array.
array_fill()	Fills an array with values.
array_pop()	Deletes the last element of an array.
array_push()	Inserts one (or) more elements to the end of an array.

Date Functions :

Function	Description
array()	Creates an array.
checkdate()	Validates a Gregorian date.
date()	Formats a local time/date.
time()	Returns the current time as a Unix timestamp.
getdate()	Returns an array that contains date and time information for a Unix timestamp.
idate()	Formats a local time/date as integer.

Static Variables : Like C, PHP supports declaring local function variables as static. These kind of variables remain in tact in between function calls, but are still only accessible from within the function they are declared. Static variables can be initialized, and this initialization only takes place the first time the static declaration is reached.

Here's an example for the use of static that runs initialization code the first time (and only the first time) the function is run :

```
function do_something()
{
static first_time = true;
if (first_time) {
// Execute this code only the first time the function is
¥'called
...
}
// Execute the function's main logic every time the function is
¥'called
...
}
```

6.1.6 RECURSIVE FUNCTIONS

A function calling itself more than once is call recursive function.

Example :

```
function factorial($n) {
    // Base case: factorial of 0 or 1 is 1
    if ($n == 0 || $n == 1) {
        return 1;
    } else {
        // Recursive call: factorial of $n is $n multiplied by
factorial of ($n-1)
        return $n * factorial($n - 1);
    }
}
// Calculate the factorial of 5
$result = factorial(5);
echo "Factorial of 5 is: " . $result; // Output: Factorial of 5 is:
120
```

In this example, the **factorial()** function calculates the factorial of a number using recursion. It checks for a base case where the input is 0 (or) 1 and returns 1. For any other input, it recursively calls itself with **(n-1)** as the argument and multiplies the current number **n** with the factorial of **(n-1)**. The recursion continues until the base case is reached.

6.2 OBJECT ORIENTATION IN PHP

Object-Oriented PHP can ease the transition from procedural to object-oriented programming. Adopting object-oriented techniques when developing your PHP scripts and applications can immediately inject greater flexibility and easier maintenance into your development environment. Programs become easier to extend (or) debug, and sharing code with other developers on your team (should you have one) becomes simpler.

6.2.1 PHP CLASSES

Object : Object Oriented Programming (OOP) is a programming concept that treats functions and data as objects. Simply put, an object is a bunch of variables and functions all lumped into a single entity. The object can then be called rather than calling the variables (or) functions themselves. Within an object there are methods and properties. The methods are functions that manipulate data within the object. The properties are variables that hold information about the object.

Class : The class contains the methods and properties (or) the characteristics of the object. It defines the object. Some examples to see how it all pieces together. We will use a vehicle as our object. All vehicles share similar characteristics, eg: number of doors, they are painted some color, they each have a price. All vehicles do similar things also, drive, turn left, turn right, stop etc. These can be described as functions (or) in OOP parlance, **methods**. So, the class holds the definition, and the object holds the value. You declare class in PHP by using the **class** keyword.

6.2.2 CREATING OBJECTS IN PHP

The symbol & in php, used for the reference of an object.

To get the desired result in PHP 4, we need to use the &(reference) operator :

```
<?php
class Foo {
   var $name;
   function Foo() {
      $this->name = "I'm Foo!\n";
   }
}
```

```
$foo = new Foo;
echo $foo->name;
$bar = &$foo;
$bar->name = "I'm Bar!\n";
// In ZE1, we would expect "I'm Foo!"
echo $foo->name;
?>
```

Output :

```
I'm Foo!
I'm Bar!
```

6.2.3 CALLING MEMBER FUNCTIONS

In PHP, member functions (also known as ***methods***) can be called on objects of a class. Here's an example of calling member functions in PHP :

```
class MyClass {
    public function sayHello() {
        echo "Hello!";
    }
    public function calculateSum($a, $b) {
        return $a + $b;
    }
}
// Create an object of MyClass
$obj = new MyClass();
// Call the sayHello() method
$obj->sayHello(); // Output: Hello!
// Call the calculateSum() method and store the result in a
variable
$result = $obj->calculateSum(3, 4);
echo $result; //
```

Output :

```
7
```

In this example, we define a class called **MyClass** with two member functions: **sayHello()** and **calculateSum()**. To call these functions, we create an object of the class using the **new** keyword and assign it to the variable **$obj**. We can then use the object to call the member functions using the arrow (->) operator, followed by the function name and any required arguments.

Calling **$obj->sayHello()** will output "Hello!", and calling **$obj->calculateSum(3, 4)** will return the sum of the two numbers, which is then echoed as "7".

6.2.4 INHERITANCE

Inheritance, in object oriented terms, allows us to create subclasses of a given class. Inheritance is a mechanism that extends and existing class. Inheriting a class would mean creating a new class with all functionality of the existing class in addition to some more. The created class is called as a ***subclass of the parent class.*** Parent is a keyword which we use to access members of a parent class. Inheritance is usually defined by using the keyword **"Extend"**.

The syntax for creating an inherited class is :

```
Class new_class_name extends original_class_name{}
```

Example :

```
<?php
  class human{
function human($hcolor)
{
    $this->hcolor=$hcolor;
}
var $legs=2;
function report()
{
return  "This <b>".get_class($this)."</b> has <b>" .$this-
```

```
>hcolor. "</b> hair,and <b>" .$this->legs. "</b> legs<br>" ;
}
}
class African extends human
{
}
//instantiate the class
$jude = new human("black");
$jane = new african("brown");
$jude->legs++; //increase legs by one
echo $jane->report();
echo $jude->report();
?>
```

6.2.5 FUNCTION OVERRIDING

The following program illustrates the method of overriding :

```
- <?php
 class Rectangle {
  public $height;
  public $width;
  public function __construct($width, $height) {
   $this->width = $width;
   $this->height = $height;
  }
  public function getArea() {
    return $this->height * $this->width;
  }
 }
 class Square extends Rectangle {
  public function __construct($size) {
    $this->height = $size;
    $this->width = $size;
  }
```

```
    public function getArea() {
        return pow($this->height, 2);
    }
}
$obj = new Square(7);
$a = $obj->getArea();
echo "$a";
?>
```

6.2.6 CREATING A CLASS CONSTRUCTOR

In PHP, a class constructor is a special method that is automatically called when an object of a class is created. It is used to initialize the object's properties or perform any setup tasks.

Here's an example of creating a class constructor in PHP :

```
class MyClass {
    public $name;
    // Constructor method
    public function __construct($name) {
        $this->name = $name;
        echo "Object created. Name: " . $this->name;
    }
}

// Create an object of MyClass and pass a name to the constructor
$obj = new MyClass("John");
// Output: Object created. Name: John
// Access the property set by the constructor
echo $obj->name; //
```

Output :

```
John
```

In this example, we define a class called **MyClass** with a constructor method **__construct()**. The constructor takes a parameter **$name** and assigns it to the object's property **$name**. Inside the constructor, we echo a message to indicate that the object has been created and display the assigned name.

To create an object of **MyClass**, we use the **new** keyword followed by the class name and pass a name argument to the constructor. This will automatically invoke the constructor and initialize the object. We can then access and use the object's properties, such as **$obj->name**, which in this case will output "John".

6.3 FORMS IN PHP

One of the most popular ways to make a web site interactive is the use of forms. With forms you can have users register for different things, submit any information you can imagine, upload files and all types of other things.

The following *programs* illustrate the use of form to send data to the php file :

PROGRAM

form1.html

```
<HTML>
<HEAD>
<TITLE>A simple form</TITLE>
</HEAD>
<BODY>
<FORM method="POST" action="form.php">
<INPUT type="submit" name="mybutton" value="Click me!">
</FORM>
</BODY>
</HTML>
```

PROGRAM

form.php

```
<HTML>
<HEAD>
<TITLE>Doing something with the form</TITLE>
</HEAD>
<BODY>
<?
if(isset($_POST['mybutton'])) {
   echo "Great! You clicked the button!\n";
} else {
   echo "Hmm...you must have come to this script without
   clicking the button.\n";
}
?>
</BODY>
</HTML>
```

6.3.1 GLOBAL AND ENVIRONMENTAL VARIABLES

Global Scope : Global scope refers to any variable that is defined outside of any function. They can be accessed from any part of the program that is not inside a function. To access a global variable from within a function, you can call it into the function with the global keyword.

```
global &varto;
```

Environment Variables : PHP has a collection of *environment variables*, which are system defined variables that are accessible from anywhere inside the PHP code, inside of functions (or) out.

Some of the Environment Variables are :

$HTTP_SERVER_VARS[ ]

Contains information about the server and the HTTP connection.

$HTTP_COOKIE_VARS[ ]

Contains any cookie data sent back to the server from the client. Indexed by cookie name.

$HTTP_GET_VARS[ ]

Contains any information sent to the server as a search string as part of the URL.

$HTTP_POST_VARS[ ]

Contains any information sent to the server as a POST style posting from a client form.

$HTTP_POST_FILES[ ]

Contains information about any uploaded files.

$HTTP_ENV_VARS[ ]

Contains information about environmental variables on the server.

6.3.2 GET AND POST METHODS

1. **Get Method :** The built-in $_GET function is used to collect values from a form sent with method="get". Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

PROGRAM

form.html

```
<form action="welcome.php" method="get">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

Program: welcome.php

```
<html>
<body>
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
</body></html>
```

2. **Post Method :** The built-in $_POST function is used to collect values from a form sent with method="post". Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

PROGRAM

form1.html

```
<form action="welcome1.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

PROGRAM

welcome1.php

```
<html>
<body>
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
</body></html>
```

6.3.3 SCRIPT TO ACCESS USER INPUT

The web application collects information from the user by means of HTML forms and processes it. Some of the information collected from users and stored at the web site is sensitive information, making security a major issue. PHP provides features that enable you to collect information from the user and to secure the information.

Example :

register.php

```
<html>
<head><title>Register</title></head>
<body>
<h1>Registration</h1>
<form method="get" action="register.php">
<table>
<tr><td>E-mail address:</td>
<td>
<input type='text' name='email'/>
</td></tr>
<tr><td>First name:</td>
<td><input type='text' name='first_name'/></td></tr>
<tr><td>Last name:</td>
<td><input type='text' name='last_name'/></td></tr>
<tr><td>Password:</td>
<td>
<input type='password' name='password'/>
</td></tr>
<tr>
<td colspan='2'>
<input type='submit' name='register' value='Register'/>
</td>
</tr>
</table>
</form>
<?php
if (!isset ($_POST['register']) ||($_POST['register'] !=
'Register')) {
?>
```

```
<?php
} else {
?>
E-mail: <?php echo $_POST['email']; ?><br />
Name: <?php echo $_POST['first_name']. ' '.
$_POST['last_name'];
?><br />
Password: <?php echo $_POST['password']; ?><br />
<?php
}
?>
</body>
</html>
```

6.3.4 ACCESSING INPUT FROM VARIOUS ELEMENTS OF A FORM

The program given below illustrates accessing input from various elements of form.

PROGRAM

form.php

```
<?php
<FORM method="POST" action="form1.php">
Fill in the text box: <INPUT type="text" name="mytextbox" size="20"><BR>
Check me or not: <INPUT type="checkbox" name="mycheckbox" value
="1"><BR>
Choose one :
Blue <INPUT type="radio" name="myradio" value="blue" CHECKED>
Green <INPUT type="radio" name="myradio" value="green">
Yellow <INPUT type="radio" name="myradio" value="yellow"><BR>
<TEXTAREA name="mytextarea" rows="4" cols="30"></
TEXTAREA><BR>
?>
```

PROGRAM

form1.php

```
<?php
echo "<BR>You typed <b>" . $_POST['mytextbox'] . "</
b> in the textbox.\n";
if(isset($_POST['mycheckbox'])) {
   echo "<BR>You checked the checkbox.\n";
} else {
   echo "<BR>You didn't check the checkbox.\n";
}
echo "<BR>The color you picked was" . $_POST['myradio'] . "\n";
echo "<BR>In the textarea, you typed:" . $_POST
['mytextarea'] . "\n";
?>
```

6.3.5 FILE UPLOADING IN PGHHHP

With PHP, it is possible to upload files to the server.

PROGRAM

RR.html

```
<html>
<body>
<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" size="60" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```

WARNING IF ANYBODY CAUGHT WILL BE PROSECUTED

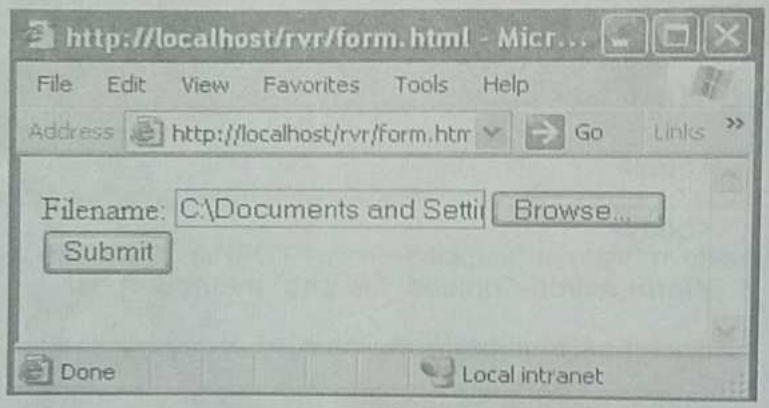
PROGRAM

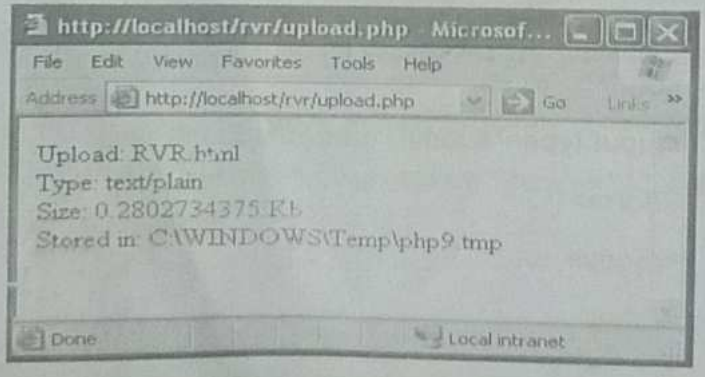
Session refers to a temporary and interactive communication b/w user's browser and a website server.

upload_file.php

```
<?php
if ($_FILES["file"]["error"] > 0)
 {
 echo "Error: " . $_FILES["file"]["error"] . "<br />";
 }
else
 {
 echo "Upload:" . $_FILES["file"]["name"] . "<br />";
 echo "Type: " . $_FILES["file"]["type"] . "<br />";
 echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
 echo "Stored in: " . $_FILES["file"]["tmp_name"];
 }
?>
```

Output :





6.4 SIGNIFICANCE COOKIE AND SESSION

6.4.1 SESSION AND COOKIE

Session : Websites typically use session cookies to ensure that you are recognised when you move from page to page within one site and that any information you have entered is remembered. For example, if an e-commerce site did not use session cookies then items placed in a shopping basket would disappear by the time you reach the checkout. You can choose to accept session cookies by changing the settings in your browser.

Cookie : 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.

6.4.2 SETTING (OR) CREATING COOKIE IN PHP

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

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

Syntax :

```
setcookie(name, value, expiration);;
```

1. **Name :** The name of your cookie. We will use this name to later retrieve your cookie, so don't forget it!
2. **Value :** The value that is stored in your cookie. Common values are username(string) and last visit(date).
3. **Expiration :** The date when the cookie will expire and be deleted. If you do not set this expiration date, then it will be treated as a session cookie and be removed when the browser is restarted.

Example for Creation of Cookie :

```
<?php
$expire=time()+5;
setcookie("user", "Rajendra Reddy", $expire);
?>
<html>
<body>
welcome to php
</body>
</html>
```

6.4.3 RETRIEVING AND DELETING COOKIE

Example for Retrieve a Cookie :

```
<html>
<body>
<?php
if (isset($_COOKIE["user"]))
  echo "Welcome" . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>
</body>
</html>
```

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

Example :

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

Note : Set the expiration date to one hour ago[time()-3600]

6.4.4 CREATING SESSION COOKIE

Without cookies, websites and their servers have no memory. A cookie, like a key, enables swift passage from one place to the next. Without a cookie every time you open a new web page the server where that page is stored will treat you like a completely new visitor.

Websites typically use session cookies to ensure that you are recognised when you move from page to page within one site and that any information you have entered is remembered. For example, if an e-commerce site did not use session cookies then items placed in a shopping basket would disappear by the time you reach the checkout. You can choose to accept session cookies by changing the settings in your browser.

6.4.5 SESSION FUNCTION IN PHP

The following are the some session functions in PHP.

- **session_cache_expire :** Return current cache expire.
- **session_cache_limiter :** Get and/or set the current cache limiter.
- **session_commit :** Alias of session_write_close.
- **session_decode :** Decodes session data from a string.
- **session_destroy :** Destroys all data registered to a session.
- **session_encode :** Encodes the current session data as a string.
- **session_get_cookie_params :** Get the session cookie parameters.
- **session_id :** Get and/or set the current session id.
- **session_is_registered :** Find out whether a global variable is registered in a session.
- **session_module_name :** Get and/or set the current session module.

- **session_name** : Get and/or set the current session name.
- **session_regenerate_id** : Update the current session id with a newly generated one.
- **session_register** : Register one (or) more global variables with the current session.
- **session_save_path** : Get and/or set the current session save path.
- **session_set_cookie_params** : Set the session cookie parameters.
- **session_set_save_handler** : Sets user-level session storage functions.
- **session_start** : Initialize session data.
- **session_unregister** : Unregister a global variable from the current session.
- **session_unset** : Free all session variables.
- **session_write_close** : Write session data and end session.

BOARD DIPLOMA EXAMINATION

MID SEM-I

WEB DESIGNING

For Computers, Final Year

Time : 1 Hour Max. Marks : 20

PART - A 4 × 1 = 4

*Note : Answer **All** the questions. Each question carries **One** mark.*

1. Define Webpage.
2. Define HTML.
3. List the attributes of Table Tag.
4. Define CSS.

PART - B 2 × 3 = 6

Note : *Answer **One** question each from 5 and 6. Each question carries **Three** marks.*

5. (a) Explain briefly any three Basic HTML Tags.

   (or)

   (b) List the steps involved in website development.

6. (a) Explain Form tag with syntax and example.

   (or)

   (b) How do you use External CSS in HTML.

PART - C 2 × 5 = 10

Note : *Answer **One** question each from 7 and 8. Each question carries **Five** marks.*

7. (a) Create a HTML page using Formatting tags.

   (or)

   (b) Explain about table tag and its attributes.

8. (a) Explain how to create frames using rows and column attributes.

   (or)

   (b) Create internal style sheet by applying Colors and Background property.

BOARD DIPLOMA EXAMINATION

MID SEM-II

WEB DESIGNING

For Computers, Final Year

Time : 1 Hour **Max. Marks :** 20

PART - A 4 × 1 = 4

Note : *Answer **All** questions. Each question carries **One** mark.*

1. Define XML.
2. List out different client side and Server side scripting languages.
3. Write syntax to Create an array.
4. Write an Example to define a function and call a function.

PART - B 2 × 3 = 6

Note : *Answer **One** question each from 5 and 6. Each question carries **Three** marks.*

5. (a) What is XML element give an example.

   (or)

   (b) Write syntax and example on conditional statements used in javascript.

6. (a) List the differences between client and server side scripting.

   (or)

   (b) Write an example on creating a function.

PART - C 2 × 5 = 10

Note : *Answer **One** question each from 7 and 8. Each question carries **Five** marks.*

7. (a) Explain XML namespace.

   (or)

   (b) Write a javascript program on recursion.

8. (a) Explain DOM.

   (or)

   (b) Write a javascript program on Array methods.

12. (a) How HTML and PHP can be combined.

(or)

(b) Write PHP script to illustrate use of static variables.

PART - C $4 \times 5 = 20$

Note : *Answer* ***Four*** *questions. Each question carries* ***Five*** *marks.*

13. (a) Explain various presentation formatting tags in HTML with an Example.

(or)

(b) Explain file uploading in PHP.

14. (a) Write the differences between IIS,PWS and APACHE.

(or)

(b) Explain the process of creating and deleting cookies in PHP with examples.

15. (a) Explain various loop statements in PHP.

(or)

(b) Explain various string functions in PHP. Write PHP script to find length of a string.

16. (a) Explain Setting or creating a cookie and session variable in PHP with an example.

(or)

(b) Write a PHP code for creating, inserting, and deleting the data in a database table.

BOARD DIPLOMA EXAMINATION

SEM END EXAMINATION

WEB DESIGNING

For Computers, Final Year

Time : 2 Hours Max. Marks : 40

PART - A 8 × 1 = 8

*Note : Answer **All** questions. Each question carries **One** mark.*

1. Define a Tag.
2. List the methods of web site maintenance.
3. How do you change color of web page in HTML.
4. List the applications of XML.
5. List comparison operators in java script.
6. Mention the names of different objects used in Java script.
7. What are magic functions in PHP.
8. How to delete cookie in PHP.

PART - B 4 × 3 = 12

Note : *Answer **Four** questions. Each question carries **Three** marks.*

9. (a) List the steps in building a web site ?

   (or)

   (b) Define CSS and List the features of CSS.

10. (a) State the need of client side scripting language.

    (or)

    (b) Explain MARQUEE tag and list all its attributes.

11. (a) List any two differences between client side and server side scripting languages.

    (or)

    (b) Mention the conditional statements in Java script.