Unit - II

2.1 Introduction: What is PHP?

PHP is a an Open Source scripting language. It is primarily used for dynamic web applications. PHP takes its syntax from C, Java and Perl language. PHP was written in the C programming Language by Rasmus Lerdorf in 1994, this was known as PHP 2. In 1998, PHP 3 was released, which was the first widely used PHP version.

PHP stands for PHP:Hypertext Preprocessor. PHP script is executed in the server and returns plain HTML as the result to the client. It is saved with the extension ".php". A PHP file can have HTML tags, CSS elements, JavaScript along with the PHP Script.

Basic Syntax of PHP

A PHP script is enclosed within a special PHP script tag for execution.

Syntax:

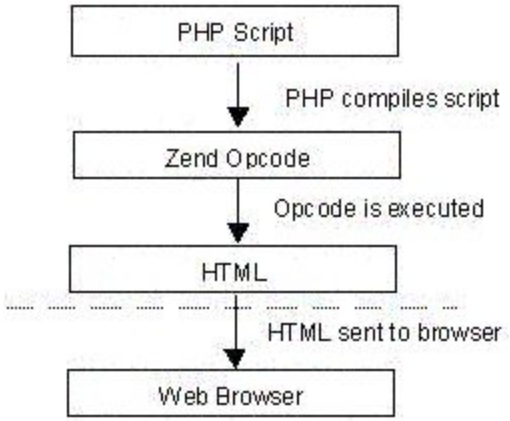
```
<?php
        php script
?>
```

Example:

```
<?php
    //Single Line Comment
    /*frist Program
    explain
    syntax of PHP*/
    echo "Hello World";
    #echo "First Program";
?>
```

Programming In Web Environment

PHP needs to be placed on a web server, so that it can interpret the script and generate the necessary HTML output. PHP's central core, that is the language engine is called as "Zend". PHP refers to the complete system. Zend Engine is used internally by PHP as a compiler and runtime engine. PHP Scripts are loaded into memory and compiled into Zend opcodes. These opcodes are executed and the HTMLgenerated is sent to the client.

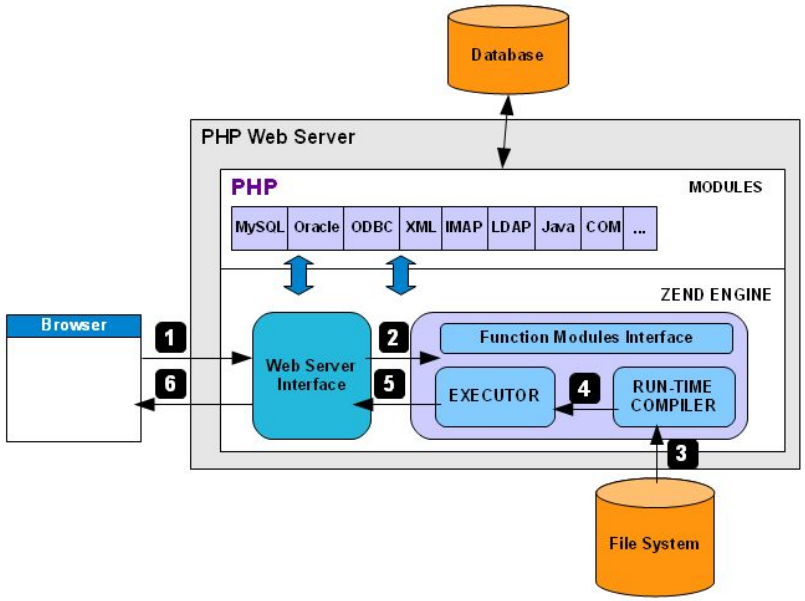


Zend engine is named after its developers Zeev and Aandi. It is reliable, has high performance and can be extended. To run a PHP script at any web server three components are needed.

1. An interpreter to analyse the PHP Script

2. A functionality section of the interpreter, which implements the functionality of PHP.
3. An Interface section that talks to the web server.

Zend Engine handles Components 1 and partially Component 2. PHP handles the remaining part of Component 2 and Component 3.



PHP Server Engine Architecture

Step 1). A request from the browser arise to execute a script on the server.

Step 2). The request is processed by the Web Server Interface and passed on to the file system.

Step 3). The requested script is loaded from the file system onto the runtime compiler for compilation. The Zend engine parses the script and generates the opcode for exectuion. During opcode generation the Zend engine includes the modules needed for the execution of the current script.

Step 4). The executor executes the opcode to generate the HTML file.

Step 5). The generated HTML file is passed onto the web server interface.

Step 6). The Web Server Interface sends the resulting HTML file to the browser.

Benifits of PHP

- No Execution in browser - The script is executed on the server and only the resulting HTML page is sent to the browser.
- Code is Invisible - The PHP Script is visible only on the server and not on the browser. This is because the script is executes on the server.
- Browser Independent - As only the HTML result is sent from the server to the browser PHP script is not dependent on any browser.
- No binary Code is generated - PHP does not generate any binary code (exe). The only output that is generated by the PHP is HTML.
- JavaScript can be Embedded - A PHP script can have JavaScript embedded in it. It will not affect the execution of the PHP script.

Drawbacks of PHP

- Need Server - To execute a PHP script a server is needed. Without a server PHP script cannot be executed.
- Need Server Resource - As PHP is a scripting language the program is interpreted everytime it is executed. This uses more server resources.

Common PHP Script Elements

PHP script elements are the elements that are processed by the PHP server. The basic most commonly used PHP script elements are

- PHP script tag
- Comments
- Statements
- Escape Sequence

PHP Script Tag

These are the tags that enclose the PHP script that is to be interpreted by the PHP server.

Syntax:

```
<?php
        ……
        …….
?>
```

Comments

A comment is a statement that is written by the programmer to give information about the program. Text given within the comment tags are not executed by the PHP server. There are two types of commnets in PHP

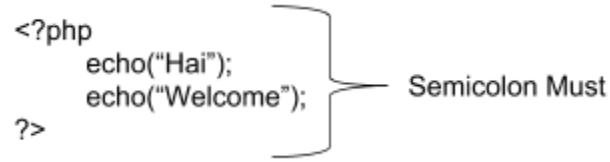
1. Single Line Comment - To make a single line as a comment in the PHP script double slash (//) or hash (#) is used.
2. Multiline Comment - To make multiple lines in a PHP script as comment the syntax used in (/*....*/).

Statements

Within the PHP script tag any number of statements can be written. Each statement must end with a semi-colon. If there is only one statement within the PHP script tag semi-colon is not necessary. If there are more than one statement within the PHP script tag semi-colon is must.

Example:

<?php echo(“hai”) ?> - semi-colon not necessary



Escape Sequence

The Escape sequence is used to add a PHP script specific character in an echo statement. For example the double quotes(“) is used to enclose the string that is to be displayed using the echo statement, if

we want to print a double quotes(“) character in the echo statement then the interpreter must be informed that the double quotes(“) that is given in the echo statement is to be printed and not processed. This is done by the use of backslash(\) also known as the escape sequence. Some of the characters that are used along with the escape sequence character are

- Double Quotes (\”)
- Single Quotes (\’)
- New Line Character (\n)
- Tab Space (\t)
- Print $ (\$)
- Print Backslash (\\)

Example

```
<?php
        #Escape Sequences
        echo "\"Hello World\" \n\t First PHP Program";
?>
```

Using Variables

Varibles are used to store information in a PHP script. To use a variable in a PHP script there are some rules to follow. The rules are called as naming rules

Naming Rules

1. All variable name must begin with a $ symbol.
2. The next character in the variable name must be an alphabet(a-z).
3. Variable names cannot have space in it.
4. Only special character allowed in a variable name is underscore(_).
5. Varibale names are case sensitive, i.e. Upper case and lower case names are different.
6. Data type of the variable is based on the value that is stored in that variable.
7. No need to declare the variable before assignment.

Example:

```
<?php
 #PHP Variables
 $aVal_ = 10;
 //b = 20; //Wrong Assignment
 echo "Variable a=".$aVal_;
 //echo "Case Sensitive $aval";
 $aVal_="Hello World";
 echo "Recent Value: $aVal_";
?>
```

Constants

A constant is a name or an identifier for a simple value. A constant value once assigned cannot be changed during the execution of the script.

Syntax:

```
define(“const_name”,value);
```

To differentiate between a variable and a constant certain naming rules are followed for the constant name.

Rules:

1. Constant Name must always be given in Upper Case.
2. Constant Name can start with an alphabet(a-z) or underscore(_).
3. Define function is used to define a constant.
4. To get the value of a constant, its name is used or constant function is used.

Example:

```
<?php
 define("PI",3.141);
 echo "PI Value = ".PI;
 echo “PI Value = ”.constant(“PI”);
?>
```

Data Types

Data Types are the basic building block of a programming language. Data Types define the type of data to be stored in a variable and the amount of memory needed to store the value. Data Types available in PHP are given below.

- String - This is used to store combination of character. String values are enclosed within double quotes (“ ”) or single quotes (‘ ’).

  Example:

```
<?php
  $str = "Hello World";
  $str1 = "Welcome To TNPT";
  echo "<b>String Value:</b> ".$str;
  echo $str1;
  //echo "<h4>String</h4><br>$str";
?>
```

- Integer - This is used to store whole numbers. When the value reaches certain limit the data type is automatically changed to float.

  Example:

```
<?php
  $intvar = 34;
  echo "<br><b>Integer Value:</b> ".$intvar;
?>
```

- Float - This is used to store fractional numbes as well as very large integer numbers.

  Example:

```
<?php
  $fltvar = 1.243;
   echo "<br><b>Float Value:</b> ".$fltvar;
?>
```

- Boolean - This is used to store TRUE or FALSE values. 0 denote FALSE value and 1 denote TRUE value.

  Example:

```
<?php
  $bool = TRUE;
  echo "<br><b>Boolaen Value: </b>".$bool;
?>
```

- NULL - This is a special data type which denotes there is no value in the variable. NULL is not equal to zero (0).

Example:

```
<?php
  $empty = NULL;
  echo “Vlaue is ”.$empty;
?>
```

- Arrays - This is used to store multiple values of same data type. In php array() function is used to create an array.

Example:

```
<?php
  $numarray = array(1,2,3,4);
  echo "2nd Array Element: ".$numarray[1];
?>
```

- Object - Objects are user defined data type. They have their own variable and fucntions.

Example:

```
<?php
  class sample
  {
    function sample()
    {
       $this->val = 10;
    }
  }
  $obj = new sample();
  echo "Value in Object: ".$obj->val;
?>
```

- Resources - These are special variables that are used to store values such as reference to a file or database table.

Operators

Operators are used to perform operations on variables and values. Operators available in PHP are given below.

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Increment/Decrement Operatos
- Logical Operators
- Conditional Operator

Arithmetic Operators

Arithmetic operators are used to perform common arithmetic operations on numeric data.

Symbol	Operator Name	Description
+	Addition	add two values
-	Subtraction	subtract two values
*	Multiplication	multiplies two values
/	Division	divides two values
%	Modulus	return remainder of dividing two values

Assignment Operators

Assignment Operators are used to assign a value to a variable in the PHP script. Variable is present on the left side of the operator(=) and the value is on the right side of the operator.

Symbol	Operator Name	Description
=	Equal	assign value from right to left side variable
+=	Add and	adds value on right to left and store in left side variable
*=	Multiply and	multiply value on right to left and store in left side variable
/=	Divide and	divide value on right to left and store in left side variable
%=	Modulus and	return remainder of value on right to left and store in left side variable

Comparison Operators

Comparison Operators are used to compare the values present in two variables or compare a value present in a variable with a value.

Symbol	Operator Name	Description
==	Double Equal	check if two values are same
!=	Not equla to	check if two values are not same
>	Greater than	check if left value is greater than right
		value
<	less than	check if eft value is less than right value
<=	less than equal to	check if left value is less than or equal to right value
>=	greater than equal to	check if left value is greater than or equal to right value

Increment/Decrement Operators

Increment/Decrement operators are used to increment or decrement the value present in a variable. Based on the position of the operator there are two types pre and post.

Symbol	Operator Name	Description
++var	Pre-Increment	Increments the value and then assigns the value
var++	Post-Increment	value is assigned and then it is incremented
--var	Pred-Decrement	Decrement the value and then assigns the value
var--	Post-Decrement	Value is assigned and then it is decremented

Logical Operators

Logical Operators are used in conditional statments to combine two conditions.

Symbol	Operator Name	Description
and, &&	and operator	return true only if both the conditions are true
or, ||	or operator	returns true if any one of the two condition is true
!	not operator	invers the result of a condition
xor	exclusive or operator	return true only if both conditions are not same

Conditional Operator

Conditional Operator is a ternary operator (it takes three operands).

syntax

var = operand1?operand2:operand3;

operand1 - It is the condition that is executed first.

operand2 - If the result of the condition is TRUE then operand2 is executed.

operand3 - If the result of the condition is FALSE then operand3 ie executed.

example

```
<?php
   $mark=65;
   $res = ($mark>45)?"Pass":"Fail";
   echo $res;
?>
```

Statements

PHP script is made up of staments. A program statement is an instruction to perform an action. Statements in PHP are classified into two types

1. Conditional Statements
2. Looping Statements

Conditional Statements

Conditional Statements are used to transfer program control based on a condition. A condition is checked in the statement and if the condition is true a set of statements are executed and if the condition is false another set of statements are executed.

There are three types of conditional statements in PHP. They are

1. if...else statement
2. if...else if statement
3. switch statement

if...else statement

If...else is a two way condition statement. It can take either TRUE part or FALSE part.

syntax:

```
if(condition)
{
        True Part
}
else
{
```

```
        False Part
    }
```

example:

```
<?php
  $mark = 39;
  if($mark>50)
  {
      echo “<B style="color: GREEN;">Pass</B>”;
  }
  else
  {
      echo “<B style="color: RED;">Fail</B>”;
  }
?>
```

if...else if statement

To check for multiple conditions in a sequential manner if...else if statement is used. In this statement if the first condition is false then another condition is checked in the else part and this continues.

syntax:

```
    if(condition 1)
    {
        True Part
    }
    else if(condition 2)
    {
        True Part
    }
    else
    {
        False Part
    }
```

example:

```
<?php
    $mark = 84;
    if($mark>=80)
    {
     echo "<b>A Grade</b>";
    }
    else if($mark>=70 and $mark<=79)
    {
     echo "<b>B Grade</b>";
    }
    else if($mark>=60 && $mark<=69)
    {
     echo "<b>C Grade</b>";
    }
    else if($mark>=50 and $mark<=59)
    {
     echo "<b>D Grade</b>";
    }
    else
    {
     echo "<b>Fail</b>";
    }
?>
```

Switch Statement

Switch statement is a value based selection statement also called as multiway branching statement. In PHP for switch statement value can have number and string.

syntax:

```
swithc(option)
{
 case value1:
        statement;
        break;
case value2:
        statement;
        break;
case valueN:
        statement;
        break;
default:
        statement;
        break;
}
```

Example:

```
<?php
  $mark = 77;
  $opt = (int)($mark/10);
  switch($opt)
  {
        case 10:
            echo "<b>S Grade</b>";
            break;
        case 9:
                echo "<b>A Grade</b>";
                break;
        case 8:
                echo "<b>B Grade</b>";
                break;
        case 7:
                echo "<b>C Grade</b>";
                break;
        case 6:
                echo "<b>D Grade</b>";
                break;
        case 5:
                echo "<b>E Grade</b>";
                break;
        default:
                echo "<b>Fail</b>";
                break;
        }
?>
```

Looping Statement

Looping statements are used to execute a set of instructions repeatedly until a certain condition is used. In PHP two types of looping statements are available, they are

1. Condition based loop
2. Iterator loop

Condition Based Loop

In condition based loops, the loop execution is controlled by a condition. If the condition is true the loop is executed else the loop is not executed. There are two categories of loops based on the place where the condition is based, they are

- Entry Contolled Loop

- Exit Controlled Loop

Entry Controlled Loop

In this loop structure the condition is placed at the beginning of the loop. If the condition is true then only the loop statements are executed. In PHP while loop and for loop are entry controlled loops

while Loop

While loop executes a set of statements as long as the specified condition is true. In general to control loop execution a loop control variable is used. This variable is initialized before the start of the while loop. Condition is checked at the start of the while loop and the loop control variable is incremented/decremented within the loop body.

syntax

```
initialization
while(condition)
{
   Loop body
   Increment/Decrement
}
```

example

```
<?php
  $val = 3;
  while($val>0)
  {
   echo "<h$val>Level $val</h$val>";
        $val--;
  }
?>
```

for loop

In for loop the initialization, condition and increment/decrement section is written in the same place. This allow the programmer to write neat and clean looping statements.

syntax

```
for(initialization;condition;increment/decrement)
{
   Loop Body
}
```

Example

```
<?php
  $mul = 3;
  $table = "<table border='2'>";
  for($i=1;$i<=5;$i++)
  {
        $res = $mul * $i;
        $table.="<tr><td>$mul x $i = </td><td> $res</td></tr>";
  }
  $table.="</table>";
  echo $table;
?>
```

In for loop when the initialization, condition and increment/decrement section is omitted it becomes an infinite loop

syntax

```
for(;;)
{
   Loop Body
}
```

Iterator Loop

PHP supports a special type of loop know as the Iterator Loop. This type of loop is used to loop through the contents of a collection such as array or objects.The loop is executed until all the elements of the collection are accessed.

syntax

```
foreach(element as itme)
{
   Loop body
}
```

Example

```
<?php
    $numarr = array(1,2,3,4,5,6,7,8,9,10);
    $sum = 0;
    foreach($numarr as $val)
    {
     echo $val."<br>";
     $sum += $val;
    }
    echo "<br><b>Sum of the Array is:".$sum."</b>";
?>
```

Working with Arrays

An array is a data structure that is used to store multiple elements of similar data type using a single variable name. There are three types of arrays, they are

- Numeric array
- Associative array
- Multidimensional array

Numeric Array

In PHP based on the index value used the array is known as numeric array. In this array only numeric values are used to access the elements in the array. The index value starts at 0 and ends at size-1, where size is the size of the array.

In PHP an array can be created using any of the following three methods

1. using []
2. using index value
3. using array function

using []

In this form of array creation size of the array is not given in the syntax. It uses only the array name and the values that are to be stored in the array within square bracket.

syntax

```
$array_name = [values];
```

example

```
$rollno = [101,102,103];
```

using index value

In this form of array creation the array is created single element at a time. Array name and index value is used to store value in the array.

syntax

```
$array_name[index] = value;
```

example

```
$rollno[0] = 101;
$rollno[1] = 102;
$rollno[2] = 103;
```

using array function

This is the safe method to create an array in PHP. In this the array() function is used to create the array.

syntax

```
$array_name = array(values);
```

example

```
$names = array("Anbu","Charan","Karthick");
```

Accessing Arrays

To access an array in PHP two methods are used

1. Using index
2. Using Iterator Loop

using index

In this numeric index value is used to access the array elements.

syntax

```
$variable = $arrayname[index];
```

using iterator loop

To access the elements in the array in an repeated manner, foreach iterator loop is used. This loop executes until all the elements in the array are read.

syntax

```
foreach($array as $value)
{
   Loop Body
}
```

Numeric Array Example

```
<html>
<head>
  <title>Arrays</title>
</head>
<body>
  <h3>Numeric Arrays</h3>
  <?php
     //Using []
     $marks = [89,92,73];
     //Using Array Index
      $marks[3] = 77;
       $marks[4] = 85;
       $marks[5] = 95;
       //Using Array Function
      $subj = array("OS","BEEE","C","Linux Lab","BEEE Lab","C Programming");
  ?>
  <p>Accessing Using Index Value</p>
  <table border="1">
  <tr><th>Subject</th><th>Mark</th></tr>
  <?php
     for($i=0;$i<6;$i++)
     {
       echo "<tr><td>".$subj[$i]."</td><td>".$marks[$i]."</td></tr>";
     }
     //Calculating Total
     $tot = 0;
     foreach($marks as $subj_mark)
     {
      $tot+=$subj_mark;
     }
     echo "<tr><td>Total</td><td><b>".$tot."</b></td></tr>";
  ?>
  </table>
</body>
</html>
```

Output

Numeric Arrays

Accessing Using Index Value

Subject	Mark
OS	89
BEEE	92
C	73
Linux Lab	77
BEEE Lab	85
C Programming	95
Total	**511**

Numeric Array Functions

PHP provides inbuilt functions that can be used on an array to perform a specific task. Some of the basic and additional array functions are listed below

Basic Array Functions

- count - used to count the number of values present in the array
  count($array_name);
- unset - used to remove an element form the array. This function uses index value to remove the element.
  unset($array[index]);
- reset - used to move the array pointer to the first element of the array.
  reset($array);
- end - used to move the array pointer to the last element of the array
  end($array);

Additonal Array Functions

- array_push - used to add an element to the last of the existing array
  array_push($array,value);
- array_pop - used to remove an element fom the last of the array
  array_pop($array);
- array_unshift - used to add an element ot the start of the existing array
  array_unshift($array,value);
- array_shift - used to remove an element from the start of the array
  array_shift($array);
- array_merge - used to merge two arrays into a single array
  array_merge($array1,$array2);
- sort - used to sort an array in ascending order
  sort($array);
- rsort - used to sort an array in descending order
  rsort($array);

Associative Array

This array functions same as that of a numeric array but the only difference is it uses string as index. It is also known as key value pair association.

In PHP an array can be created using any of the following three methods

1. using []
2. using index value
3. using array function

using []

In this form of array creation size of the array is not given in the syntax. The array name and the values to be stored in the array are given as “Key” “Value” pair within the square bracket.

syntax

$array = [“key1”=>”value”,“key2”=>”value”,“key3”=>”value”];

Example

$cse_stud = [“01”=>”Anbu”, “02”=>”Bala”, “03”=>”Guna”];

using index value

In this form of array creation the array is created single element at a time. Array name and index value is used to store value in the array.

syntax

$array_name[“index”] = value;

example

$cse_stud[“01”] = “Anbu”;
$cse_stud[“02”] = ”Bala”;
$cse_stud[“03”] = “Guna”;

using array function

This is the safe method to create an array in PHP. In this the array() function is used to create the array.

syntax

$array = array(“Key1”=>”Value”,”Key2”=>”Value”);

Example

$cse_stud = array(“01”=>”Anbu”, “02”=>”Bala”, “03”=>”Guna”);

Accessing Arrays

To access an array in PHP two methods are used

1. Using string index
2. Using Iterator Loop

using index

In this string index value is used to access the array elements.

syntax

$variable = $arrayname[“string_index”];

using iterator loop

To access the elements in the array in an repeated manner, foreach iterator loop is used. This loop executes until all the elements in the array are read.

syntax

```
foreach($array as $key=>$value)
{
   Loop Body
}
```

Associative Array Example

```
<html>
<head>
 <title>Arrays</title>
</head>
<body>
 <h2>Associative Arrays</h2>
 <?php
        //Using []
        $studDCE = ["01"=>"Anbu","02"=>"Bala"];

        //Using Array Index
        $studDEEE["01"] = "Arun";
        $studDEEE["02"] = "Mani";

        //Using Array Function
        $dept = array("101"=>"DCE","102"=>"DEEE");
 ?>
```

```
<h2>Student Details</h2>
<?php
        echo "<h3>".$dept["101"]."</h3>";
        foreach($studDCE as $roll=>$name)//Using Iterator Loop
        {
                echo "<br>Roll: ".$roll;
                echo "<br>Name: ".$name;
        }
        echo "<h2>".$dept["102"]."</h2>";
        foreach($studDEEE as $roll=>$name)
        {
                echo "<br>Roll: ".$roll;
                echo "<br>Name: ".$name;
        }
 ?>
</body>
</html>
```

Output

Associative Arrays

Student Details

DCE

Roll: 01
Name: Anbu
Roll: 02
Name: Bala

DEEE

Roll: 01
Name: Arun
Roll: 02
Name: Mani

Associative Array Functions

PHP provides inbuilt functions that can be used on an array to perform a specific task. Three important associative array functions are listed below

- array_key_exists - this function is used to check if a given string index is already present in the array or not. It will return TRUE if string index is present and FALSE if not present.
  array_key_exists("index",$array);
- array_keys - this function is used to get only the key from the array.
  array_keys($array);
- array_values - this function is used to get only the values present in the array.
  array_values($array);

Multidimensional Array

A multidimensional array is a type of array which stores another array at each index instead of single element. In general a multidimensional array is an array of arrays. To access a multidimensional array more than one indices are used. The best way to create multidimensional array is by the use of array() function.

A simple multidimensional array is two dimensional array.

syntax:

$array = array(array(elements),array(elements));

example:

$array = array(array(1,2,3),array(4,5,6));

Accessing Arrays

To access an array in PHP two methods are used

1. Using string index
2. Using Iterator Loop

using index

In this two index values are used to access the array elements. In general the two index are known as row and column.

syntax

$variable = $arrayname[row][column];

using iterator loop

To access the elements in the array in an repeated manner, foreach iterator loop is used. This loop executes until all the elements in the array are read. To access the values in a multidimensional array nested foreach loops are used.

syntax

```
foreach($array as $row)
{
   foreach($row as $val)
   {
      Loop Body
   }
}
```

Example

```
<head>
  <title>Arrays</title>
</head>
<body>
  <h2>Multi Dimensional Array</h2>
  <?php
        //Using array() function
        $ary = array(array(1,2,3,4),array(5,6,7,8));
        $val = $ary[1][1];
        echo "<br> 1Row 2Col Value: $val";
  ?>
  <p>
        <b>Using Nested Foreach Loop</b><br>
        <?php
         foreach($ary as $row)
         {
          foreach($row as $val)
          {
                echo " $val";
          }
          echo "<br>";
         }
        ?>
  </p>
</body>
</html>
```

Output

Multi Dimensional Array

1Row 2Col Value: 6

Using Nested Foreach Loop
1 2 3 4
5 6 7 8

Using Functions

A function is a block of statement that can eb used repeatedly in a program. A function can have parameters which can be used to send

values to the function to process. In PHP a function is executed only when it is called.

In PHP there are two parts that are needed to execute a function properly. They are

1. Creating a PHP function
2. Calling a PHP function

Creating a PHP function

Before executing a function in PHP it must be created. To create a function four parts are needed in PHP.

1. Valid name for the function
2. "function" keyword before the function name
3. parameter list after the function name
4. Function code block within curly braces.

syntax

```
function Function_Name(argument list)
{
   Function Code block;
}
```

Calling a PHP function

After creating a function to execute it the function is called using the function name with parameter list.

syntax

```
Function_Name(argument list);
```

Example

```
<html>
<head>
  <title>Function</title>
</head>
<body>
  <h2>Simple Example</h2>
  <?php
          //Creating function
          function display()
          {
           echo "Welcome";
          }
  ?>
  <h4><?php display(); ?></h4>
  <h3><?php display(); ?></h3>
</body>
</html>
```

Output

Simple Example

Welcome

Welcome

Function Types

Based on the types of parameters and returning values there are 5 types of functions available in PHP. They are

1. No Parameter
2. With Parameter
3. Call by Reference
4. Default Value
5. Return Value

No Parameter

In this type of function, there is no parameters passed to the function.

syntax

```
function function_name()
{
   Code
}
```

function call

```
function_name();
```

With Parameter

In this type of function, parameters are passed to the function for processing.

syntax

```
function function_name($para1, $para2)
{
   Code
}
```

function call

```
function_name($para1, $para2);
```

Example

```
<?php
    $a = 10;
    $b = 20;
    //Function with Arguments
    function swap($x, $y)
    {
      $temp = $x;
      $x = $y;
      $y = $temp;
      echo "<label>After Swapping</label>";
      echo "<br>a = ".$x;
      echo "<br>b = ".$y;
    }

    //Calling Function
    swap($a,$b);
?>
```

Call by reference

In this type of function the values are passed to the function by reference. By passing the value by reference the changes done on the function will reflect on the calling place. A parameter is passed by reference by adding an ampersand (&) to the variable name in the function definition.

syntax

```
function function_name(&$para1, &$para2)
{
   Code
}
```

function call

```
function_name($para1, $para2);
```

example

```
<?php
    $a = 10;
    $b = 20;
    echo "<label>Before Swapping</label>";
    echo "<br>a = ".$a;
    echo "<br>b = ".$b;
    swap($a,$b);
    echo "<br><label>After Swapping</label>";
    echo "<br>a = ".$a;
    echo "<br>b = ".$b;
```

```
//Call by reference
function swap(&$x, &$y)
{
 $temp = $x;
 $x = $y;
 $y = $temp;
}
?>
```

Default Value

In this function, the parameters are assigned a default value in the function itself. If the function call doesnot have any parameter the value provided in the function is automatically taken.

syntax

```
function function_name($para1 = value)
{
   Code
}
```

function call

```
function_name($para1);
        (or)
function_name();//This call automatically takes the value
                provided in the function
```

example

```
<?php
//Default Value
function AddImg($img = "noimg.jpg")
{
 echo '<img src="'.$img.'" >';
}
AddImg("01.jpg");
AddImg("02.png");
AddImg();
?>
```

Return Value

To return a value from the function the **return** statement is used. In this type of function once the return statement is reached the function execution stops and sends the value back to the calling place. Only one value can be sent using the return statement.

syntax

```
function function_name(parameter List)
{
   Code
   return $val;
}
```

function call

```
$variable = function_name(parameter list);
```

Example

```
<?php
function fun($x,$y,$z)
{
 $val = 0;
 while($x && $y && $z)
 {
  $x--;$y--;$z--;
  $val++;
 }
 return $val;
}
$res = fun(10,5,8);
echo "<br>Result: ".$res;
```

?>

OOP

Object Oriented Programming is an approach to software development that model real world applications. In PHP object oriented concepts was introduced in PHP5, this is helpful in creating complex and reusable web applications. In PHP most important Object Oriented concepts are

- Class
- Memmber Variable
- Member Function
- Object
- Inheritance

Class

A class is sn user defined data type, which has its own variables and functions. In general a class is a template for making instance of the same kind.

Member Varaiable

These are the variables defined inside a class. These variables are not accessible outside of the class if they are declared as private or protected.

Member Function

These are the functions that are defined inside a class and are used to access the data present in the object.

Object

Object is an individual instance of the data type defined by a class. Using a class more than one object can be created.

Syntax

```
class class_name
{
   var $variable1;
   var $variable2;
   function function_name(parameter list)
   {
         Function Code
   }
}
```

Object creation

```
$obj = new class_name;
        (or)
$obj = new class_name();
```

To access the member variables and member functions of a class outside of the class, object name and access operator (->)is used.

syntax

```
$obj->variable;
$obj->function_name();
```

To access the member variables and member functions of a class within the class, $this keyword and access operator (->) is used.

syntax

```
$this->variable;
$this->function_name();
```

Example

```
<?php
        class student
        {
         var $roll;
         var $name;
         function set_data($r,$n)
         {
                $this->roll = $r;
                $this->name = $n;
         }
         function get_data()
         {
                echo "<br>Roll No: ".$this->roll;
                echo "<br>Name: ".$this->name;
         }
        }

  $stu1 = new student;
  $stu2 = new student();
  echo "<br> Two Objects created";

  $stu1->set_data(101,"Anbu");
  $stu2->set_data(102,"Bala");

  $stu1->get_data();
  $stu2->get_data();
 ?>
```

Output

```
Two Objects created
Roll No: 101
Name: Anbu
Roll No: 102
Name: Bala
```

Inheritance

It is the process of making the member variables and member functions of one class available in another class. The class whose variables and functions made available to another class is called as the base class. The class that access the variables and functions is called as derived class. The derived class can have its own functions and variables in addition to the base class variable and functions. In php “extends” keyword is used to implement inheritance.

syntax

```
class base
{
   var $basevariable;
   function base_function(parameter_list)
   {
         Function Code
   }
}
class derived extends base
{
   var $derivedvariable;
   function derived_function(parameter_list)
   {
         Function Code
   }
}
```

Example

```
<?php
        //Base Class
        class student
        {
         var $roll;
         var $name;
         function set_data($r,$n)
         {
                $this->roll = $r;
                $this->name = $n;
         }
         function get_data()
         {
                echo "<br>Roll No: ".$this->roll;
                echo "<br>Name: ".$this->name;
         }
        }

        //Derived Class
        class alumni extends student
        {
         var $year_pass;
         function set_yearpass($yr)
         {
                $this->year_pass = $yr;
         }
         function get_yearpass()
         {
                echo "<br>Year of Pass: ".$this->year_pass;
         }
        }

        $stu1 = new student;
        $stu2 = new alumni;
        echo "<br> Two Objects created";

        $stu1->set_data(101,"Anbu");
        $stu2->set_data(102,"Bala");
        $stu2->set_yearpass(2017);

        $stu1->get_data();
        $stu2->get_data();
        $stu2->get_yearpass();
 ?>
```

In the above example the base class “student” details can be accessed in the derived class “alumni”.

Access Modifiers

In object oriented programming access modifiers restrict the places in which a member variable and member function can be accessed within a PHP Script.

There are three Access Modifiers available in PHP

1. Public
2. Private
3. Protected

Public

This is the default access specifier that is assigned by PHP for the members of a class. A public member can be access anywhere within the script.

Example

```
 <?php
        class base
        {
          var $x;
```

```
    function set_data($a)
    {
        $this->x = $a;
    }
    function get_data()
    {
        echo "<br>Value in X: ".$this->x;
    }
  }
  $obj = new base;
  $obj->x = 10;
  $obj->get_data();
?>
```

In the above example the Member variable x and member function get_data both are public, so they can be accessed outside the class using the object name.

Private

A private member is accessible only within that class and cannot be accessed outside of the class.

Example

```
<?php
    class base
    {
     private $x;
     function set_data($a)
     {
        $this->x = $a;
     }
     function get_data()
     {
        echo "<br>Value in X: ".$this->x;
     }
    }
  $obj = new base;
  //$obj->x = 20; // This access is not allowed
  $obj->set_data(20);
  $obj->get_data();
?>
```

In the above example the member variable of the class is declared as private. In the script it is not possible to access the variable using the object of the class. It can be accessed only within the class.

Protected

A protected member is accessible within the class in which it is declared and also the class that inherits it. It is not accessible outside.

Example

```
<?php

    class base
    {
     protected $x;
    }

    class derived extends base
    {
     var $y;
     function set_data($a,$b)
     {
        $this->x = $a;
        $this->y = $b;
     }
     function get_data()
     {
        echo "<br>Value in X: ".$this->x;
        echo "<br>Value in Y: ".$this->y;
```

```
    }
  }
$obj = new derived;
//$obj->x = 20;
$obj->set_data(20,10);
$obj->get_data();
?>
```

In the above example, the member variable x of the base class is declared as private, so it is accessible only in that class and the class that inherits it. By this the class derived can access the variable x.

Constructor

A constructor is a special type of function which is called automatically when an object is created for the class. It is executed only once. It is used to allocate resources.

syntax

```
function __construct(Parameter_List)
{
   Constructor Code
}
```

Destructor

A destructor is called as soon as there are no other reference to a particular object in the PHP script. It is used to deallocate the resources.

Syntax

```
function __destruct()
{
   Destructor Code
}
```

Example

```
<?php
 class employee
 {
  var $name;
  var $dept;
  function __construct($n,$d)
  {
       echo "<br>Calling Constructor<br>";
       $this->name = $n;
       $this->dept = $d;
  }
  function display()
  {
       echo "<br>Employee Name: ".$this->name;
       echo "<br>Department: ".$this->dept;
  }
  function __destruct()
  {
       echo "<br><br>Calling Destructor";
  }
 }
 $obj = new employee("Sajid","HR");
 $obj->display();
?>
```

Output

```
Calling Constructor

Employee Name: Sajid
Department: HR

Calling Destructor
```

String Manipulation

These are the functions used to manipulate string in PHP. In PHP strings are case sensitive. There are many string manipulation function available in PHP, some of the important functions are listed below.

- Length
- Reverse
- SubString
- String Search
- String Replace
- Case Change
- Comparison
- Explode/Implode

Length

This function is used to get the length of the given string. It takes only one parameter, the string and it will return a number indicating the length of the string

syntax

$length = strlen($string);

Reverse

This function is used to reverse a given string. It takes the string as the parameter and returns the reversed string.

syntax

$reverse = strrev($string);

SubString

This function is used to get a sub string from the given string. It takes three parameters, the original string, starting position for the substring and the length of the substring. If length is not given the substring is formed by extracting upto the end of the original string.

Syntax

$substring = substr($string,Start,Lenght);

String Search

In PHP there are two functions that perform string search

1. strstr
2. strpos

The strstr function is used to search for a string within a string, if the string is found then the function returns the string from the matching point. In this function if the before parameter is set to TRUE then the string before the matching point is returned.

Syntax

$search = strstr($string,$search,before);

The strpos function is used to search for a string within a string and return the index of the first occurence of the string.

Syntax

$position = strpos($string,$search);

String Replace

This function is used to find a string within a string and replace it with a new string. It has an additional parameter count which counts the number of occurence the string has been replaced.

Syntax

str_replace($search,$replace,$string,$count);

Case Change

In PHP there are a set of functions that are used for changing the case of a string. There are four types of case change functions

1. Uppercase
2. Lowercase
3. First Letter Uppercase
4. Title Case

The Uppercase function changes all the characters in the string to uppercase.

Syntax

strtoupper($string);

The Lowercase function changes all the characters in the string to lowercase.

Syntax

strtolower($string);

The First Letter Uppercase function changes the first character in the string to uppercase.

Syntax

ucfirst($string);

The Title Case function changes the first character of each word in the string to uppercase.

syntax

ucwords($string);

Comparison

In PHP to compare two strings there are two functions available. They are

1. strcmp
2. similar_text

The strcmp function is used to compare two strings and based on the comparison it will return a numeric value.

Syntac

strcmp($string1,$string2)

0 - if both the string are equal

<0 - if string1 is smaller than string2

>0 - if string1 is larger than string2

Regular Expression

In PHP regular expression provide a general patter to perform pattern matching. This uses various operators to create the general pattern, these operators are called as meta characters. Each meta character has its own meaning. Some of the most common meta characters and their meaning are given below

Meta Character	Meaning
/	use for begining and ending of a pattern
^	Matching at the begining
$	Matching at the end
.	Check only one element
[Char List]	Single Character in the list
[^Char List]	Single Character not in the list
elm*	0 or many times
elm+	1 or maby times

elm?	0 or 1 time
{}	Exact number of Count
\	Check for meta character

To perform the pattern match with a string PHP uses the preg_match function. This function return 1 if the string matches with the pattern else it will return 0.

Syntax

preg_match(“Pattern”,$string);

Some example of using pattern matchin in PHP is given below

Check Only One Element

Example

```
$str = "a";
$res = (preg_match("/^.$/",$str)?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Valid

Check Character in a List

Example

```
$str = "G";
$res = (preg_match("/[abcd]$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Invalid

Check Elements 0/many, 1/many

Example

```
$str = "11";
$res = (preg_match("/^10*1$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Valid

Example

```
$str = "11";
$res = (preg_match("/^10+1$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Invalid

Check Elements 0/1, count

Example

```
$str = "1001";
$res = (preg_match("/^10?1$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Invalid

Example

```
$str = "10001";
$res = (preg_match("/^10{3}1$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern Match Valid

Check Meta Character

Example

```
$str = "1021.01";
$res = (preg_match("/^[1-9][0-9]+\.[0-9]{2}$/",$str))?"Valid":"Invalid";
echo "Pattern Match $res";
```

Output

Pattern match Invalid

2.2 File and Directory Handling

PHP supports various file and directory handling functions. These functions are used to perform file operations and directory related functions on the server.

File Handling

In PHP file handling is performed by the following four operations.

1. Open File
2. Read File
3. Write File
4. Close File

Open File

In PHP to open a file fopen function is used. It takes two arguments as parameter, first is the file name and second is the file opening mode. If the file is opened correctly this function returns a file handle.

syntax

```
$file_handle = fopen(filename,File Opening mode);
```

File Opening Mode

There are six different file opening modes available in PHP. Each mode has its own property.

File Opening Mode	Description
r	Opens the file in read only mode.
w	Opens the file in write only mode and deletes the existing content in the file. If the file doesnot exist it creates a new file.
a	Opens the file in write only mode and places the pointer at the end of the file. The existing content is not deleted. If the file doesnot exist it creates a new file.
r+	Opens the file in reading and writing mode.
w+	Opens the file in reading and writing mode. This deletes the existing content of the file. If the file doesnot exist it creates a new file.
a+	Opens the file in reading and writing mode. Places the pointer at the end of the file. If the file doesnot exist it creates a new file.

example

```
$f_handle = fopen("test.txt","r");
```

Read file

To read a file in PHP fread function is used. It takes two parameters, the file handle and file size. File handle is obtained from fopen function and the size of the file is obtained using filesize function.

syntax

```
$size = filesize($filename);
$text = fread($file_handle,$size);
```

example

```
$size = filesize("text.txt");
$text = fread($f_handle,$size);
```

File Write

To write content in a file PHP uses fwrite function. This function takes two parameters, file handle and the text to be written on the file.

syntax

```
fwrite($file_handle,text);
```

example

```
fwrite($f_handle,"Welcome");
```

File Close

In PHP fclose function is used to close the file. It takes only one parameter, the file handle. It is always necessary to close the file that is opened to properly save it content.

Syntax

```
fclose($file_handle);
```

Example

```
fclose($f_handle);
```

File Handling Complete Example

```
<?php
    $text = "File Handling Introduction";
    $fhandle = fopen("test.txt","w");
    fwrite($fhandle,$text);
    fclose($fhandle);
    $fhandle = fopen("test.txt","r");
    $size = filesize("test.txt");
    $text = fread($fhandle,$size);
    echo "File Content";
    echo "<br>$text";
?>
```

Directory Handling

Directories are containers for files. PHP provides a set of directory related functions and class. Directory handling functions are used to open and access the directories present in the server.

Directory Handling Functions

PHP supports three functions for directory handling. To open a directory opendir function is used. To read the contents of the directory readdir function is used, this function is used in a looping structure so that it can read the content one by one. To close the opened directory closedir function is used.

Syntax

```
$dir_handle = opendir(path);
$file_nmae = readdir($dir_handle);
closedir($dir_handle);
```

Example

```
<?php
  $dh = opendir("images");
  while(($file=readdir($dh))!=FALSE)
  {
  echo "<br>$file";
 }
 closedir($dh);
?>
```

Dir Class

The dir() function is a object based function that creates an instance of the directory class. This function takes the path of a directory and returns an object of that directory. PHP uses read() function to read the contents of the object. To close the opened directory close() function is used.

Syntax

```
$obj = new dir(Directory path);
$obj->read();
$obj->close();
```

Example

```
<?php
  $obj = dir("images");
  while(($file=$obj->read())!=FALSE)
  {
  echo "<br>$file";
 }
 $obj->close();
?>
```

Including Files

PHP allows the content of one file to be loaded into another file. This process is done before execution begins at the server. There are two functions that can be used for file inclusion

- include() function
- require() function

The include() function copies all the text of the given file into the current file. If there is any error in loading the file then this function generates a warning and the script will continue execution.

The require() function also performs the same operation. The difference is if there is any error in loading the file then this function generates a fatal error and stop the execution of the script

Syntax

```
include(Include File);
require(Include File);
```

Example

menu.php

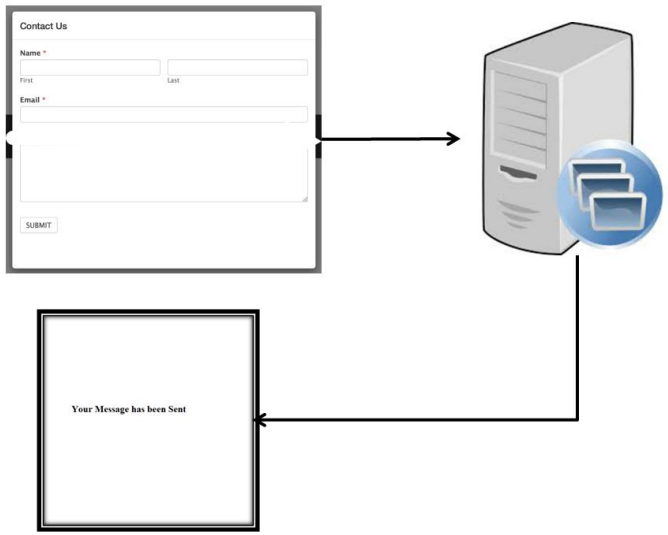
```
<a href="eg30.php">Patter Matching</a><br>
<a href="eg31.php">File Handling</a><br>
<a href="eg32.php">Directory Handling</a>
```

include.php

```
<html>
<head>
 <title>File Inclusion</title>
</head>
<body>
 <h3>File Inclusion</h3>
 <?php
   include("menu.php");
 ?>
 <h3>This Page Include Menu.php</h3>
<!-- <?php
   include("test.php");
 ?>
 <h3>Testing require</h3>-->
</body>
</html>
```

2.3 Working with Forms

In web based programming a form is an interface between the user and the server. Without this interface it is not possible to send the data that the user has entered in a website to the server. In general Form is an html element, it collects the data entered in the web page and it is send to the server for processing.



Form Process Flow

Processing Forms

HTML form tag has its own attributes for collecting and sending data to the server.

Syntax

```
<form action="Script_URL" method="POST/GET">
    HTML input elements to collect Data with name attribute
    Submit button to send data to "Script_URL"
</form>
```

Example

```
<form action="<?php echo $_SERVER["PHP_SELF"] ?>"
method="POST">
    <input type="text" name="stname"/>
    <input type="submit" value="Send name"/>
</form>
```

The form tag has two main attributes, action and method. Action Attribute is used to specify the file on the server to which the collected data is to be sent. It may point to another script file on the server or it may point to the same file. To point the same file "$_SERVER['PHP_SELF']" is used in the action tag.

The second attribute is the method attribute. Method Attribute specifies the method in which the data is to be sent to the server. It used URL encoding schema to send the data. URL encoding uses name, value pair (name=value) method to send data, different pairs are separated by ambersent(&) symbol. It can send the data using two methods GET or POST.

GET

In this method the data is appended to the page specified in the action attribute using question mark (?) and the data is using URL encoding method.

Example

localhost/test.php?rollno=101&last=mani

In this above example the data rollno and name is sent to the PHP script page test.php using URL encoding method. The limitations of the GET method is that it can send only 1024 characters of data, the data that is sent is visible in the browser address bar and this method is not used to send sensitive data such as username/password. In PHP the data sent using GET method is processed using the $_GET associative array. By providing the name that is used as the array index the value is accessed.

Example

```
$roll_no = $_GET[“rollno”];
$stud_name = $_GET[“name”];
```

POST

This method send the data collected form the webpage using HTTP header. It used URL encoding to encode the data and send it through the header called QUERY_STRING. The main advantages of POST method is that there is no restriction on the size of data to be sent, the data is not visible to the user and it can send binary data. In PHP the data sent using POST method is processed using $_POST associative arary. By providing the name that is used as the array index the value is accessed.

Example

```
$roll_no = $_POST[“rollno”];
$stud_name = $_POST[“name”];
```

Form Validation

Form validation is the process of checking the inputs given by the user on a form. This is done so that only correct data is processed and also no input is left empty. In PHP the validation is done on the server side. The data is sent to the server and in the server the correctness of the data entered is checked. If the data is inconsistent then the server intimates the webpage about it.

PHP server side validation is performed by the following steps.

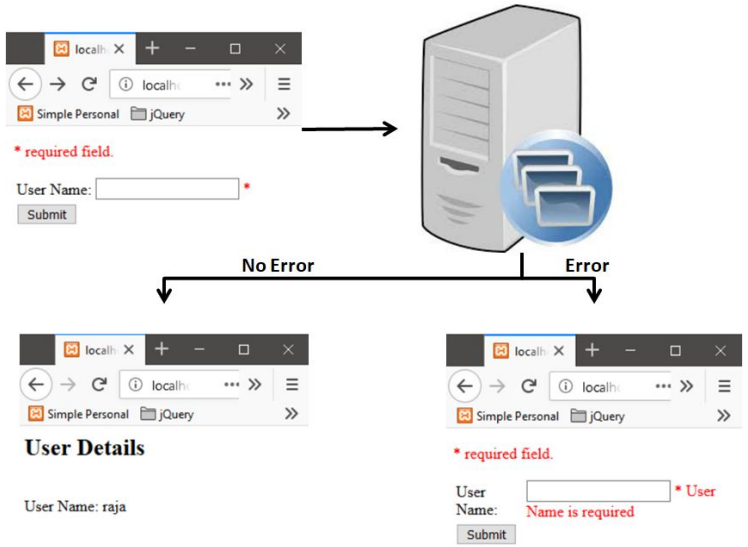
- Check if data is sent from the client to the server. This is done using the isset() function.
- Check if the submitted data has value in it. This is done using empty() function.
- If the data is inconsistent then an error message is assigned to the input tags. This is done by adding a span element to all the input elements. To highlight the error message the span elements are assigned an error class in CSS.

Example

```
<html>
<style>
      .error
       {
        color:#FF0000;
       }
</style>
<?php
 $nameerr = ""; // define variables and set to empty values
 if (isset($_POST["usrname"])
```

```
  {
  if (empty($_POST["usrname"]))
  {
   $nameerr = "User Name is required";
  }
  else
  {
   $usrname = $_POST["usrname"];
  }
?>
<body>
<?php
  if(empty($nameerr))
  {
     echo "<h2>User Details</h2>";
     echo "<br>User Name: ".$usrname;
  }
}//End of If ISSET
else
{
?>
<form method = "post" action = "<?php echo $_SERVER["PHP_SELF"];?>">
  <label>User Name:</label>
     <input type = "text" name = "usrname">
     <span class = "error">* <?php echo $nameerr;?></span><br>
     <input type = "submit" name = "submit" value = "Submit">
 </form>
<?php
}//End of Else ISSET
?>
</body>
</html>
```

Output



Introdution to advanced PHP Concepts

Advanced topics that are covered in PHP are

- Error Handling
- Cookies
- Session

Error Handling

Error handling is the process of effectively managing the errors that occur during the program runtime. In PHP error handling done using two methods, by using the die() function or by using the try...catch block.

die() function

The die function is used to display an error message and exit the execution of the program. This is the simplest form of error handling option available in PHP

syntax

```
die("Error Message");
```

Example

```
<?php
    $mark = -10;
    if($mark<0)
      die("Invalid Mark");
    if($mark>50)
       echo "Pass";
    else
       echo "Fail";
  echo "Marks Displayed";
?>
```

In the above example the mark is given in negative value which is an error. An if statement is used to check if the mark value is less than 0 and call the die() function. Once the die() function is called it will display the error message given in it and exit the script execution.

try...catch

This method of error hadling is also known as exception handling. This is an effective way of error handling, in this if an error occurs then it displays an error message and the script is not exited. The script continues to execute.

syntax

```
try
{
   throw new Exception(msg);
}
catch(Exception $e)
{
   Display error Message
}
```

Example

```
<?php
 $mark = -10;
 try
 {
        if($mark<0)
          throw new Exception("Invalid Mark");
        if($mark>50)
          echo "Pass";
        else
          echo "Fail";
 }
 catch(Exception $ex)
 {
        echo $ex->getfile();
        echo " Error: ".$ex->getmessage();
 }
 echo "<br>Marks Displayed";
?>
```

In the above example even after the error the script executes fully.

Cookies

Cookies are text files that are stored on the client compute. They are kept for tracking purpose. In general a cookie us used to identify a user. There are three major functions that can be performed on a cookie.

- Create cookie
- Access cookie
- Delete cookie

Create cookie

To create a cookie in PHP setcookie function is used. This function creates the cookie with the given parameters and stores the cookie in the client computer.

syntax

setcookie(name,value,expiry,path,domain,security);

- name - It is the variable name used to access the cookie
- value - This is the value that is stored in the cookie. It is accessed using the name parameter.
- expiry - This provides the lifetime of the cookie. It time value that is expressed in seconds.
- path - This describes the webpage in which the cookie is to be made available. If "/" is used then the cookie is made available for the whole website.
- domain - This is used to specify the domain for which the cookie is valid.
- security - This parameter takes 0 or 1 as value. 0 denotes HTTP and 1 denotes HTTPS.

Access Cookie

To access a cookie in the PHP script $_COOKIE associative array format is used. In this the index value is the name used in the cookie creation process.

syntax

$_COOKIE["name"];

Delete Cookie

To delete a cookie the setcookie() function is used with a negative expiry value.

syntax

setcookie("name","",time()-10,"/","",0);

Example

setcookie.html

```
<html>
<body>
 <form method="POST" action="cookie.php">
  <input type="text"name="usrname"/>
  <input type="submit" value="set cookie"/>
 </form>
</body>
</html>
```

cookie.php

```
<?php
if(isset($_POST["usrname"]))
{
  $usrname = $_POST["usrname"];
  setcookie("usrname",$usrname, time()+30,"/","",0);
}
```

```
?>
<html>
<body>
  <p>User Name stored in Cookie for 30 seconds</p>
</body>
</html>
```

getcookie.php

```
<html>
<body>
  <?php
    $val = $_COOKIE["usrname"];
  ?>
  <p>User Name in Cookie is "<?php echo $val ?>"</p>
</body>
</html>
```

In the above example, the file setcookie.html page collects the user name and sends it to the cookie.php script. The PHP script creates a cookie with expiry time as 30 seconds and stores the value in the client computer. The getcookie.php page access the cookie using the name of the cookie and the value stored in the cookie is displayed. If the getcookie.php page is executed after 30 seconds then the user name cookie will not be available.

Session

A session is a way to store information to be used across multiple pages. A session creates a file in a temporary directory on the server where registered session variables and their values are stored. There are two main functions that can be performed using a session

- Create session
- Destroy session

Create Session

To create a session first the session_start() function is called in the PHP. This function starts a session where values can be stores on the server. To store values on the current session $_SESSION associative array format is. In this the index value is the name used to identify the value that is stored

Syntax

```
session_start();
$_SESSION[“Name”] = Value;
```

Destroy Session

There are two methods available to destroy a session from the server. The first method uses session_destroy() function which deletes all the session variables that are stored in the current session. The second method uses unset() function, this is used to delete a single session variable form the current session.

Syntax

```
session_destroy();
unset($_SESSION[“Name”]);
```

Example

setsession.html

```
<html>
<body>
 <form method="POST" action="session.php">
  <label>Name: </label>
  <input type="text"name="studname"/><br>
```

```
  <name>Roll No: </name>
  <input type="text" name="rollno"/>
  <input type="submit" value="set Session"/>
 </form>
</body>
</html>
```

session.php

```
<?php
if(isset($_POST["studname"]))
{
  $stuname = $_POST["studname"];
  $rollno = $_POST["rollno"];
  session_start();
  $_SESSION["studname"] = $stuname;
  $_SESSION["rollno"] = $rollno;
}
?>
<html>
<body>
  <p>Student Name and Roll No stored in session</p>
</body>
</html>
```

getsession.php

```
<html>
<body>
  <?php
    session_start();
    $studname = $_SESSION["studname"];
    $rollno = $_SESSION["rollno"];
    unset($_SESSION["studname"]);
    session_destroy();
  ?>
  <p>Student Name is "<?php echo $studname ?>" Roll No is
 <?php echo $rollno ?> </p>
</body>
</html>
```

In the above example setsession.html gets student name and roll number and send the data to the session.php script. In session.php a session is created and student name, roll number are stored in that session. The getsession.php page is used to access the student name and roll number stored in the session variable. After reading the values the session variables are deleted using unset() and session_destroy() functions.

Review Questions

Part-A

1. What is PHP?
2. What is a Variable in PHP?
3. What is a constant in PHP?
4. What is a conditional Statement in PHP?
5. What is a looping statement in PHP?
6. What is an array in PHP?
7. Write the syntax of function in PHP.
8. Define class in PHP.
9. Write the syntax of any two string manipulation funcitons in PHP.
10. What are the file reading modes available in PHP?

11. What is a form?
12. What is a cookie?
13. What is a session?

Part-B

1. Write the benifits of PHP.
2. Explain the usage of variables in PHP.
3. Write about eh data types avaialeble in PHP.
4. Explain if...else statement in PHP with an example.
5. Explain switch statement in PHP with an example.
6. Write about Associative array in PHP.
7. Explain function with return value in PHP.
8. What are the access specifiers available in PHP? Explain.
9. Write a PHP script to read and write a file.
10. How form is processed in PHP? Explain with an example.

Part-C

1. Explain in detail about PHP Programming in web environment.
2. What are the common PHP script elements? Explain in detail.
3. Explain in detail about Operators in PHP.
4. What are the looping statements available in PHP? Explain with example.
5. What are the numeric array functions availabe in PHP?
6. Explain in detail about Function with parameter and function with default values in PHP.
7. Explain in detail about inheritance in PHP.
8. Explain in detail about Regular Expression in PHP.
9. Explain in detail about Including Files in PHP.
10. Write in detail about form validation in PHP.
11. Elaborate advance concepts in PHP.