Web Technology
Unit- II

Topic: PHP

Syllabus

PHP :

- Introduction to PHP, uses of PHP,
- general syntactic characteristics,
- Primitives, operations and expressions, output,
- control statements,
- arrays,
- functions,
- pattern matching,
- form handling,
- files,
- cookies, session tracking,
- using MySQL with PHP

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- String manipulation,
- Arrays & Functions,
- Form handling,
- Cookies & Sessions,
- using MySQL with PHP,

Introduction to PHP

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use
- **What is a PHP File?**
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code are executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

Introduction to PHP

- **What Can PHP Do?**
- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Features of PHP

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

PHP Syntax

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with **<?php** and ends with **?>**:

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".

sample code –

Example 1 to print Hello World using PHP

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

sample code –

Example 2 variable declaration

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

echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
```

Out Put

"Hello world!"
5
10.5

PHP Variables

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

sample code –

Example 3- To output text and a variable

```php
<?php
$txt = "W3Schools.com";
echo "I love $txt";
?>
```

```php
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . ;
?>
```

I love W3Schools.com!

Program for addition of two numbers

```
<?php
$n1=5;
$n2=6;
$sum=$n1+$n2;
Echo “Summation is”.$sum;
?>
```

PHP Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - local
  - global
  - static

PHP Variables Scope-
Global and Local Scope

Example

```
<?php
$x = 5; // global scope

function myTest() {
  echo "Variable x inside function is: $x";
}

myTest();

echo "Variable x outside function is: $x";
?>
```

Output

Variable x inside function is:

Variable x outside function is: 5

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

PHP Variables Scope-
Global and Local Scope

Example

```
<?php
function myTest()
 {
 $x = 5; // local scope
 echo "Variable x inside function is: $x";
}

myTest();
echo "Variable x outside function is: $x";
?>
```

Output

```
Variable x inside function is: 5
Variable x outside function is:
```

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

PHP Variables Scope-
The global Keyword

Example

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();  // run function
echo $y; // output the new value for
variable $y
?>
```

Output

```
15
```

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

PHP Comparison Operators

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

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- **conditions & Loops,**
- Functions,
- String manipulation,
- Arrays & Functions,
- Form handling,
- Cookies & Sessions,
- using MySQL with PHP,

Conditions and Loops

Conditional Statements

- If else
- Elseif ladder
- Switch case

Loop Statements

- While
- Do while
- For
- Foreach

If…else

Syntax

```
If(condition)
    statements;
else
   statements;
```

Example

```
<?php
$n=5;
If($n%2 == 0)
Echo “Number is Even”;
Else
Echo “Number is Odd”;
?>
```

Elseif

Syntax

```
If(condition)
    statements;
Elseif(condition)
    statements;
Else
    statements;
```

Example

```
<?php
$day=date("l");
If($day == "Saturday"))
Echo "Happy Weekend";
Elseif($day == "Sunday"))
Echo "Happy Sunday";
Else
Echo "Nice Working day";
?>
```

Switch case

Syntax

```
Switch(expression)
{
Case constant_expression:
        statements;
         break;

Default:
        statements;
}
```

Example

```
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
        default:
        echo "Your favorite color is neither red, blue!";
} ?>
```

While Loop

Syntax

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

Example

```
<?php
$x = 1;

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

Do-While Loop

Syntax

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

Example

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

For Loop

Syntax

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

Example

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

Foreach Loop

Syntax

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

Example

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

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- **Functions,**
- String manipulation,
- Arrays & Functions,
- Form handling,
- Cookies & Sessions,
- using MySQL with PHP,

User Defined Functions

Syntax

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

Example

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

Parameterized Functions

Example

```
<?php
function Add($a,$b) {
   $sum=$a+$b;
      echo “Sum is $sum";
}
Add(10,20);
?>
```

Output

Sum is 30

Returning value through function

Example

```
<?php
function Add($a,$b) {
   $sum=$a+$b;
      return $sum;
}
$Result=Add(10,20);
echo “Sum is $Result";
?>
```

Output

Sum is 30

Setting default values for function parameter

Example

```
<?php
function Add($a,$b=300)
  {
    $sum=$a+$b;
   echo “Sum is $sum";
  }

 Add(10);
 Add(10,20);
 ?>
```

Output

```
Sum is 130
Sum is 30
```

Dynamic Function Calls

Example

```
<?php
 function Hello() {
 echo "Hello How R U?";
  }
 $fh = "Hello";
 $fh();
  ?>
```

Output

```
Hello How R U?
```

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- **String manipulation,**
- Arrays & Functions,
- Form handling,
- Cookies & Sessions,
- using MySQL with PHP,

String Manipulation

String Function	Description	Example
strlen()	returns the length of a string	<?php echo strlen("Hello world!"); ?> // outputs 12
str_word_count()	counts the number of words	<?php echo str_word_count("Hello world!"); ?> // outputs 2
strrev()	reverses a string	<?php echo strrev("Hello world!"); ?> // outputs !dlrow olleH
strpos()	searches for a specific text within a string.	<?php echo strpos("Hello world!", "world"); ?> // outputs 6
str_replace()	replaces some characters with some other characters in a	<?php echo str_replace("Ram", "Holly", "Hello Ram"); ?>

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- String manipulation,
- **Arrays & Functions**,
- Form handling,
- Cookies & Sessions,
- using MySQL with PHP,

Arrays

Create an Array in PHP

- In PHP, the array() function is used to create an array:
  - array();
- In PHP, there are three types of arrays:
  - **Indexed arrays** - Arrays with a numeric index
  - **Associative arrays** - Arrays with named keys
  - **Multidimensional arrays** - Arrays containing one or more arrays

Arrays- Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically as below :
  - $cars = array("Volvo", "BMW", "Toyota");
- The index can be assigned manually:
  - $cars[0] = "Volvo";
    $cars[1] = "BMW";
    $cars[2] = "Toyota";

Arrays- Indexed Arrays

- **To create and print array**

  Example-

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

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

  Example

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

Arrays- Indexed Arrays

Loop Through an Indexed Array

Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

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

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


    foreach($cars as $a) {
    echo $a;

} ?>
```

Arrays- Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:
- Method -1

```
$age = array("Peter"=>"35", "Ben"=>"37");
```

- Method- 2

```
$age['Peter'] = "35";
$age['Ben'] = "37";
```

Arrays- Associative Arrays

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

- **Output:**
- Peter is 35 years old.

Loop Through an Associative Array

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {
    echo "Key=" . $x . "Value=" . $x_value;
    echo "<br>"; } ?>
```

- **Output:**
- Key=Peter, Value=35
  Key=Ben, Value=37
  Key=Joe, Value=43

Arrays- Multidimensional Arrays

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep.

PHP - Two-dimensional Arrays: A two-dimensional array is an array of arrays

First, take a look at the following table:

Name	Stock	Sold
Volvo	22	18
BMW	15	13

Example

```
$cars = array  (
  array("Volvo",22,18),
  array("BMW",15,13)  );
```

Functions on Array

- **print_r() –**Prints all elements of an array in standard format
- **extract()-**Converts array into variables
- **compact()-**Converts group of variables into array
- **is_array() –**to check weather a particular elements is an array or not.
- **sort()** - sort arrays in ascending order
- **rsort()** - sort arrays in descending order,
- **asort()-**sorts associative arrays in **ascending order**, on **values**
- **ksort()-**sorts associative arrays in **ascending order**,on **keys**
- **arsort()-** sorts associative arrays in **descending order**,on **values**
- **krsort()-**sorts associative arrays in **descending order**,on **keys**

Functions on Array: print_r()

Example

```
<?php
$a =
array ('a' => 'apple', 'b' => 'banana‘)
print_r ($a);
?>
```

Output

```
Array
    (
    [a] => apple
    [b] => banana
    )
```

Functions on Array: extract()

Example

```
<?php
$my_array = array(“Rno" => “1",
“ Name" => “Anna",
 “Class" => “TE Comp");
extract($my_array);
echo  $Rno;
echo $Name;
echo  $Class;
?>
```

Output

```
1
Anna
TE Comp
```

Functions on Array: compact()

Example

```
<?php
$firstname = "Peter";
$lastname = "Griffin";
$age = "41";

$result = compact("firstname", "lastname", "age");

print_r($result);  ?>
```

Output

```
Array
(
[firstname] => Peter [lastname] => Griffin
 [age] => 41
)
```

Functions on Array: sort()

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);           //sort in ascending order
print_r ($numbers);
rsort($numbers);        //sort in descending order
print_r ($numbers);
?>
```

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- String manipulation,
- Arrays & Functions,
- **Form handling,**
- Cookies & Sessions,
- using MySQL with PHP,

Form Handling- using Post Method

a.html

```
<form action="welcome.php" method="post">
Name:
 <input type="text" name="t1">
<br>
<input type="submit">
</form>
```

Welcome.php

```
Welcome
 <?php
echo $_POST["t1"];
?>
```

Form Handling- using Get Method

a.html

```
<form action="welcome.php" method="get">
Name:
 <input type="text" name="t1">
  <input type="submit">
</form>
```

Welcome.php

```
Welcome
 <?php
echo $_GET["t1"];
?>
```

Form Handling-

Difference between get and post method

$_GET is an array of variables passed to the current **script via the URL parameters.**

$_POST is an array of variables passed to the current script **via the HTTP POST method.**

When to use GET?

Information sent from a form with the GET method is **visible to everyone**. GET also has limits on the amount of information to send. The limitation is about 2000 characters.

When to use 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.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- String manipulation,
- Arrays & Functions,
- Form handling,
- **Cookies & Sessions,**
- using MySQL with PHP,

Cookie

- **What is a 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.
- **Create Cookies With PHP-** is created with the setcookie() function.

- **Syntax**
- setcookie(*name, value, expire, path, domain, secure, httponly*);
- Only the *name* parameter is required. All other parameters are optional.

Cookie-Create/Retrieve a Cookie

```
<?php
setcookie(“name”, “Amit”, time() + (86400 * 30), "/");  // 86400 = 1 day
?>


<?php
if(isset($_COOKIE[“name”]))
 {
  $nm=$_COOKIE[“name”];
  echo “Hello”,$nm;
}
else { echo “Coocke is not set”;    } ?>
```

Cookie-Modifying a Cookie

To modify a cookie, just set (again) the cookie using the setcookie() function:

Cookie- Deleting Cookies

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- **What is a PHP Session?**
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.

Sessions- Start a Session

- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.
- **Example -** demo_session1.php

```
<?php
session_start(); // Start the session  ?>
<html>
<body>
<?php
$_SESSION["favcolor"] = "green"; // Set session variable
echo "Session variables are set."; ?>
</body>
</html>
```

Sessions: Get Session Variable Values

- Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").
- session variable values are stored in the global $_SESSION variable
- **Example-** demo_session2.php
  - ```
    <?php
    session_start(); ?>
    <html>
    <body>
    <?php
    // Echo session variables that were set on previous page
    echo "Favorite color is " . $_SESSION["favcolor"];
    ?>
    </body>
    </html>
    ```
    ```

Sessions- Modify a Session

To change a session variable, just overwrite it:

```
<?php
session_start();  ?>
<html>
<body>
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body>
</html>
```

Sessions: Destroy a Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example-

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

<html>
<body>

<?php
session_unset();     // remove all session variables
session_destroy();    // destroy the session
?>

</body>
</html>
```

Outline

- Introduction to PHP,
- Features,
- sample code,
- PHP script working,
- PHP syntax,
- conditions & Loops,
- Functions,
- String manipulation,
- Arrays & Functions,
- Form handling,
- Cookies & Sessions,
- **Using MySQL with PHP,**

Open a Connection to MySQL

```
<?php
// Create connection
$conn = new mysqli(“localhost”, “root”, “”);
//MySQLi extension (the "i" stands for improved)

// Check connection
 if(!$conn){
      die('Could not connect: '.mysqli_connect_error());
    }
    echo 'Connected successfully<br/>';
?>
```

Select Data From a MySQL Database

```
<?php
   $conn = mysqli_connect(‘localhost’, ‘root’, ‘root’, ‘db1’);
    if(!$conn){
      die(mysqli_connect_error());
    }
    echo 'Connected successfully<br>';

    $sql = 'SELECT * FROM STUD';
    $rs=mysqli_query($conn, $sql);
    $nrows= mysqli_num_rows($rs); ?>
```

Select Data From a MySQL Database

```
if($nrows > 0) {
 while($row = mysqli_fetch_assoc($rs)){
    echo "ID :{$row['id']}  <br>";
    echo    "FNAME : {$row['firstname']} <br>";
    echo    "LNAME : {$row['lastname']} <br>";
    echo    "------------------------------<br>";
 }
}
else {  echo " No result";  }
mysqli_close($conn);
?>
```