Unit 1:- PHP Basics

- ✓ PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- ✓ PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.(Active server page)...
- ✓ PHP -7 is the latest version of PHP.
- ✓ **Php is a object oriented programming language** , it is used to create the object…
- ✓ It means any items, which have property and method..

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
?>
```

1. The default file extension for PHP files is ".php".
2. html =.html,css=.css java script=.js or html ,,,,php=.php
3. A PHP file normally contains HTML tags, and some PHP scripting code.
4. built-in PHP function "echo" to output the text "Hello World!" on a web page:

5) Echo is used to show the out put …or give message..same as printf in c language…

What is 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 for ex wamp...
- ❖ PHP is free to download and use.

What is a PHP File?

I. PHP files can contain text, HTML, CSS, JavaScript, and PHP code

II. PHP code is executed on the server, and the result is returned to the browser as plain HTML

III. PHP files have extension ".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.

Why PHP?

a. PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)

b. PHP is compatible with almost all servers used today (Apache, IIS, etc.)

c. PHP supports a wide range of databases

d. PHP is free. Download it from the official PHP resource: www.php.net

e. PHP is easy to learn and runs efficiently on the server side

example :-

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "My first PHP script!";
?>
</body>
</html>
```

Echo Is same as printf,,,,,to show the output……….or message…

```
<!DOCTYPE html>
<html>
<body>
<h1>My first PHP page</h1>
<?php
echo "Hello World!";
?>
</body>
</html>
```

PHP Data Types

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

PHP supports the following data types:

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

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

Example

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

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

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = "Hello world!";

$y = 'Hello world!';


echo $x;

echo "<br>";

echo $y;

?>


</body>

</html>
```

PHP Integer

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

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

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

Example

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

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = 5985;

var_dump($x);

?>


</body>

</html>
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

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

Example

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

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

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

</body>
</html>
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

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

PHP Array

An array stores multiple values in one single variable.

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

Example

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

```
<!DOCTYPE html>

<html>

<body>


<?php

$cars = array("Volvo","BMW","Toyota");

var_dump($cars);

?>


</body>

</html>
```

PHP Object

An object is a data type which stores data and information on how to process that data.

In PHP, an object must be explicitly declared.

First we must declare a class of object. For this, we use the class keyword. A class is a structure that can contain properties and methods:

Example

```
<?php
class Car {
  function Car() {
    $this->model = "VW";
  }
}
```

```
// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

Try it Yourself »

```
<!DOCTYPE html>

<html>

<body>



<?php

class Car {

  function Car() {

    $this->model = "VW";

  }

}

// create an object

$herbie = new Car();



// show object properties

echo $herbie->model;

?>



</body>

</html>
```

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

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

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = "Hello world!";

$x = null;

var_dump($x);

?>


</body>

</html>
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

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

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Show it

+	Addition	$x + $y	Try it »
-	Subtraction	$x - $y	
*	Multiplication	$x * $y	Try it »
/	Division	$x / $y	
%	Modulus	$x % $y	Try it »
**	Exponentiation	$x ** $y 5^2	

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	
x = y	x = y	Try it »
x += y	x = x + y	
x -= y	x = x - y	Try it »
x *= y	x = x * y	
x /= y	x = x / y	Try it »
x %= y	x = x % y	

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	
==	Equal	$x == $y	Try it »
===	Identical	$x === $y	

!=	Not equal	$x != $y	Try it »
<>	Not equal	$x <> $y	
!==	Not identical	$x !== $y	Try it »
>	Greater than	$x > $y	
<	Less than	$x < $y	Try it »
>=	Greater than or equal to	$x >= $y	
<=	Less than or equal to	$x <= $y	Try it »
<=>	Spaceship	$x <=> $y	

PHP Increment / Decrement Operators

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

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

Operator	Description	
++$x	Increments $x by one, then returns $x	Try it »
$x++	Returns $x, then increments $x by one	
--$x	Decrements $x by one, then returns $x	Try it »
$x--	Returns $x, then decrements $x by one	

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name

And	And	Try it »
Or	Or	
Xor	Xor	Try it »
&&	And	
||	Or	Try it »
!	Not	

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Result	
.	Concatenation	Concatenation of $txt1 and $txt2	Try it »
.=	Concatenation assignment	Appends $txt2 to $txt1	

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	
+	Union	$x + $y	Try it »

==	Equality	$x == $y	
===	Identity	$x === $y	Try it »
!=	Inequality	$x != $y	
<>	Inequality	$x <> $y	Try it »
!==	Non-identity	$x !== $y	

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Try it »
??	Null coalescing	$x = *expr1* ?? *expr2*	

Operator	Name
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponentiation

Unit 2 : Control Structures and Loops

PHP Conditional Statements

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

In PHP we have the following conditional statements:

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

---

PHP - The if Statement

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

Syntax

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

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

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

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

Try it Yourself »

PHP - The if...else Statement

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

Syntax

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

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

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

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

Try it Yourself »

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

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

Syntax

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

Example

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

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

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

==

```
!DOCTYPE html>

<html>

<body>


<?php

$t = date("H");


if ($t < "20") {

  echo "Have a good day!";

}

?>


</body>

</html>
```

==

```
<!DOCTYPE html>

<html>

<body>


<?php

$t = date("H");
```

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

</body>
</html>

=====================================================
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
        echo "Have a good night!";

}

?>


</body>

</html>
```

==

PHP switch Statement

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

Syntax

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

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically. The `default` statement is used if no match is found.

Example

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

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

```
    </body>

</html>
```

==

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

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

The following chapters will explain and give examples of each loop type.

The PHP while Loop

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

Syntax

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

Examples

The example below displays the numbers from 1 to 5:

```
<!DOCTYPE html>

<html>

<body>


<?php

$x = 1;


while($x <= 5) {

  echo "The number is: $x <br>";

  $x++;

}

?>


</body>

</html>
```

The PHP do...while Loop

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

Syntax

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

Examples

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

```
<!DOCTYPE html>

<html>

<body>

<?php

$x = 1;

do {

  echo "The number is: $x <br>";

  $x++;

} while ($x <= 5);

?>


</body>
```

The PHP for Loop

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

Syntax

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

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

```
<!DOCTYPE html>

<html>

<body>


<?php

for ($x = 0; $x <= 10; $x++) {

  echo "The number is: $x <br>";

}

?>


</body>

</html>
```

The PHP foreach Loop

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

Syntax

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

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

Examples

The following example will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

<!DOCTYPE html>

<html>

<body>

```
<?php
$colors = array("red", "green", "blue", "yellow");


foreach ($colors as $value) {
  echo "$value <br>";
}
?>


</body>
</html>
```

===========================////=================================

Unit 3 :- Functions, Objects and Errors

PHP Functions

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

strrev: reverse the string..

```
 <!DOCTYPE html>

<html>

<body>

<?php

echo strrev("Hello World!");

?>

</body>

</html>
```

strlen() - Return the Length of a String

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

```
<!DOCTYPE html>

<html>

<body>

<?php

echo strlen("Hello world!");

?>

 </body>

</html>
```

strpos() - Search For a Text Within a String

The PHP `strpos()` function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

```
<!DOCTYPE html>

<html>

<body>

<?php

echo strpos("Hello world!", "world");
```

```
 ?>

</body>

</html>
```

str_replace() - Replace Text Within a String

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

```
<!DOCTYPE html>

<html>

<body>

<?php

echo str_replace("world", "Dolly", "Hello world!");

?>

 </body>

</html>
```

PHP pi() Function

The `pi()` function returns the value of PI:

```
<!DOCTYPE html>

<html>

<body>

<?php

echo(pi());
```

```
    ?>

/body>

</html>
```

PHP min() and max() Functions

The `min()` and `max()` functions can be used to find the lowest or highest value in a list of arguments:

```
<!DOCTYPE html>

<html>

<body>

<?php

echo(min(0, 150, 30, 20, -8, -200) . "<br>");

echo(max(0, 150, 30, 20, -8, -200));

?>

</body>

</html>
```

PHP abs() Function

The `abs()` function returns the absolute (positive) value of a number:

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

echo(abs(-6.7));

?>


</body>

</html>
```

PHP sqrt() Function

The `sqrt()` function returns the square root of a number:

```
<!DOCTYPE html>

<html>

<body>

<?php

echo(sqrt(64) . "<br>");

echo(sqrt(0) . "<br>");

echo(sqrt(1) . "<br>");

echo(sqrt(9));

?>

</body>

</html>
```

PHP round() Function

The `round()` function rounds a floating-point number to its nearest integer:

```
<!DOCTYPE html>

<html>

<body>


<?php

echo(round(0.60) . "<br>");

echo(round(0.50) . "<br>");

echo(round(0.49) . "<br>");

echo(round(-4.40) . "<br>");

echo(round(-4.60));

?>


</body>

</html>
```

Random Numbers

The `rand()` function generates a random number:

```
<!DOCTYPE html>

<html>

<body>
```

```
<?php

echo(rand());

?>



</body>

</html>
```

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

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

Create a PHP Constant

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

Syntax

define(*name*, *value*, *case-insensitive*)

```
<!DOCTYPE html>

<html>

<body>
```

```
    <?php

// case-sensitive constant name

define("GREETING", "Welcome to W3Schools.com!");

echo GREETING;

?>


</body>

</html>

==================================

<!DOCTYPE html>

<html>

<body>

<?php

 define ("x",8);

define("PI" ,3.14);

$y=PI*x*x;

echo " the values  is= ".$y;

?>

</body>

</html>

==================================
```

PHP Constant Arrays

In PHP7, you can create an Array constant using the `define()` function.

```
<!DOCTYPE html>

<html>

<body>

<?php

define("cars", [

  "Alfa Romeo",

  "BMW",

  "Toyota"

]);

echo cars[0];

?>

</body>

</html>
```

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

```
html>

<head>

<title>

Another php example

</title>

</head>

<body>

<h1> Login Page </h1>

<form method="post" action="greate2.php">

Enter the mark  of one sub:-

    <input type="text" name="n1" />

Enter the marks  of 2sub:-

        <input type="text" name="n2" />

    <input type="submit" value="click me" onclick="greate2()" />

</form>

</body>
```

```
    </html>


================================

<html>

<head>

<title>

Another php example

</title>

</head>

<body>

<h1> Login result </h1>

<?php

    $x = $_POST["n1"];

  $y= $_POST["n2"];

            function   greate2($x,$y)

    {

      if($x>$y)

        {

           echo " gretest no is".$x;

           }

            else

           {
            echo "gretest no is".$y;

           }

  }

?>

<----------------->

<?php

        greate2($x,$y);

?>

</body>

</html>

================================

<html>

<head>

<title>

Another php example

</title>

</head>

<body>

<h1> Login Page </h1>

<form method="post" action="wagmare.php">

Enter the mark one sub:-

    <input type="text" name="n1" />
    Enter the marks 2sub:-
        <input type="text" name="n2" />
    <input type="submit" value="Login" onclick="wagmare()" />
</form>
</body>
</html>
==============================
<html>
<head>
<title>
Another php example
</title>
</head>
<body>
<h1> Login result </h1>
<?php
    $x = $_POST["n1"];
  $y= $_POST["n2"];
            function   wagmare($x,$y)
    {
      if($x>$y)
        {
```

```
        return (  $x);


          }

           else

          {


                      return  ($y);

           }

 }

?>

<?php

        $p =wagmare($x,$y);

             echo " \n the gretest no by value return is =".$p;


?>


</body>

</html>

==========================================
```

Unit 4:- working with Forms

PHP Form Handling

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

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

</body>
</html>
```

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

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

```
<html>
<body>

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

</body>
</html>
```

The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

=====================================

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

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

</body>
</html>
```

and "welcome_get.php" looks like this:

```
<html>
<body>

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

</body>
</html>
```

==

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_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** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

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

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

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

Developers prefer POST for sending form data.

====================================

Unit 5:- More with Forms

PHP Form Validation

PHP Form Validation Example

* required field

Name

E-mail:

Website

Comment:

Gender: Female Male

Your Input:

Yogesh Prabhakar Deshmukh
yogesh14deshmukh1976@gmail.com
www.dacc.edu.in
hhhh
male

he validation rules for the form above are as follows:

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

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF
"]);?>">
```

When the form is submitted, the form data is sent with method="post".

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's filter_var() function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $emailErr = "Invalid email format";
}
```

PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";


<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
```

```
     } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-
9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
```

```
   <p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF
"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

Unit 6 :- Storing and Protecting Data

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

A cookie 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.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the `isset()` function to find out if the cookie is set:

```
<!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); //
86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
     echo "Cookie '" . $cookie_name . "' is set!<br>";
     echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the
value of the cookie.</p>

</body>
</html>
```

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

Modify a Cookie Value

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

```
  <!DOCTYPE html>
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
     echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
     echo "Cookie '" . $cookie_name . "' is set!<br>";
     echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

<p><strong>Note:</strong> You might have to reload the page to see the new
value of the cookie.</p>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the setcookie() function with an expiration date in the past:

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

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

```
   ?>

</body>
</html>
```

PHP Sessions

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.

Start a PHP Session

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

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

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

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

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

</body>
</html>
```

Get PHP 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").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

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

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

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

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

Destroy a PHP Session

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

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();

echo "All session variables are now removed, and the session is
destroyed."
?>

</body>
</html>
```

Unit 7 :- MySQL Database

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

- With PHP, you can connect to and manipulate databases.

- MySQL is the most popular database system used with PHP.

- ## Create a MySQL Database Using MySQLi and PDO

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

- The following examples create a database named "myDB":

- ### Example (MySQLi Object-oriented)

-
```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

```
<html>

<head>

<title>

Another php example

</title>

</head>

<body>

<h1> MySQL Database connection  </h1>

<h2> User Login </h2>

<form method="post" action="loginuser1.php">

   UserName

        <input type="text" name="uname" /> <br>

        Password

        <input type="password" name="upass" /> <br>

   <input type="submit" value="Login" />

</form>
```

```
   <h2> User Signup </h2>
<form method="post" action="storeuser.php">
        Userid
        <input type="text" name="uid" /> <br>
   UserName
        <input type="text" name="uname" /> <br>
        Address
        <input type="text" name="uaddr" /> <br>
        City
        <input type="text" name="ucity" /> <br>
        Phone
        <input type="text" name="uphone" /> <br>
        Password
        <input type="password" name="upass" /> <br>
        Confirm Password
        <input type="password" name="upass2" /> <br>
   <input type="submit" value="Signup" />
</form>
</body>
</html>
===================
<html>
<head>
<title>
Another php example
```

```
    </title>

</head>

<body>

<h1> MySQL Database connection  </h1>

<h2> Login User </h2>

<?php


        $name = $_POST["uname"];

        $pass = $_POST["upass"];


        $con = mysql_connect("localhost:3307","root","admin") or die("Some error " .
mysql_error());

        mysql_select_db("mytest",$con);

        $result = mysql_query("select * from users where username ='".$name."' and
password='".$pass."'");

   if(mysql_num_rows($result) > 0 )

                echo "<h2> User Successfully Logged-in </h2>";

        else

                echo "<h2> User Not Logged-in </h2>";

        mysql_close($con);


?>

</body>

</html>
```

======================end ===============================

Bibliography

Learning Resources:		
1	Reference Books	1)PHP The Complete Reference :-Hozner ,Steven. 2) Black Book - Web Technology : Kogent Learning Soluction
2	Websites	**a) http://www.W3School.com** **b) http://www.tutorialpoint.com**