PHP is a recursive acronym for "PHP: Hypertext Preprocessor".

Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

PHP is an object-oriented language.

PHP files have extension ".php"

PHP Case Sensitivity

Features

* **Performance**: Script written in PHP executes much faster then those scripts written in other languages such as JSP & ASP.
* **Open Source Software**: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
* **Platform Independent**: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
* **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS etc.
* **Embedded**: PHP code can be easily embedded within HTML tags and script.

<!DOCTYPE html>

<html>

<body>

<?php

echo "My first PHP script!";

?>

</body></html>

Variable

<!DOCTYPE html>

<html>

<body>

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

echo $txt;

echo "<br>";

echo $x;

echo "<br>";

echo $y;

?>

</body>

</html>

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)

<!DOCTYPE html>

<html>

<body>

<?php

$txt = "W3Schools.com";

echo "I love $txt!";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$txt = "W3Schools.com";

echo "I love " . $txt . "!";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$x = 5;

$y = 4;

echo $x + $y;

?>

</body>

</html>

Variables Scope

PHP has three different variable scopes:

* local
* global
* static

<!DOCTYPE html>

<html>

<body>

<?php

$x = 5; // global scope

function myTest() {

// using x inside this function will generate an error

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

echo "<p>Variable x outside function is: $x</p>";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body><?php

function myTest() {

$x = 5; // local scope

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

// using x outside the function will generate an error

echo "<p>Variable x outside function is: $x</p>";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?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

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$x = 5;

$y = 10;

function myTest() {

$GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];

}

myTest();

echo $y;

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

function myTest() {

static $x = 0;

echo $x;

$x++;

}

myTest();

echo "<br>";

myTest();

echo "<br>";

myTest();

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$txt1 = "Learn PHP";

$txt2 = "W3Schools.com";

$x = 5;

$y = 4;

print "<h2>" . $txt1 . "</h2>";

print "Study PHP at " . $txt2 . "<br>";

print $x + $y;

?>

</body>

</html>

Data Types

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

String

<!DOCTYPE html>

<html>

<body>

<?php

$x = "Hello world!";

$y = 'Hello world!';

echo $x;

echo "<br>";

echo $y;

?>

</body>

</html>

Integer

<!DOCTYPE html>

<html>

<body>

<?php

$x = 5985;

var\_dump($x);

?>

</body>

</html>

Float

<!DOCTYPE html>

<html>

<body>

<?php

$x = 10.365;

var\_dump($x);

?>

</body>

</html>

Boolean

A Boolean represents two possible states: TRUE or FALSE.

$x = true;  
$y = false;

Array

<!DOCTYPE html>

<html>

<body>

<?php

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

var\_dump($cars);

?>

</body>

</html>

Object

<!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>

NULL Value

<!DOCTYPE html>

<html>

<body>

<?php

$x = "Hello world!";

$x = null;

var\_dump($x);

?>

</body>

</html>

String Functions

<!DOCTYPE html>

<html>

<body>

<?php

echo strlen("Hello world!");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

echo str\_word\_count("Hello world!");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

echo strrev("Hello world!");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

echo str\_replace("world", "Dolly", "Hello world!");

?>

</body>

</html>

# 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).

### Syntax

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

Parameters:

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

<!DOCTYPE html>

<html>

<body>

<?php

// case-sensitive constant name

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

echo GREETING;

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

// case-insensitive constant name

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

echo greeting;

?>

</body>

</html>

Constants are Global

Constants are automatically global and can be used across the entire script.

<!DOCTYPE html>

<html>

<body>

<?php

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

function myTest() {

echo GREETING;

}

myTest();

?>

</body>

</html>

Operators

* Arithmetic operators: +,-,\*,/,%
* Assignment operators:+=,-=,\*=,/=,%=
* Comparison operators:>,<,>=,<=,==
* Increment/Decrement operators:++,--
* Logical operators:and,or,&&,||,!
* String operators: .,.=

# if...else... Statements

### Syntax

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

if...elseif....else Statement

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

<!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>

# switch Statement

### 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;*  
}

<!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>

Loops

* **while**
* **do...while**
* **for**
* **foreach**

<!DOCTYPE html>

<html>

<body>

<?php

$x = 1;

while($x <= 5) {

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

$x++;

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$x = 1;

do {

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

$x++;

} while ($x <= 5);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

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

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {

echo "$value <br>";

}

?>

</body>

</html>

# Functions

### Syntax

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

<!DOCTYPE html>

<html>

<body>

<?php

function writeMsg() {

echo "Hello world!";

}

writeMsg();

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

function familyName($fname) {

echo "$fname Refsnes.<br>";

}

familyName("Jani");

familyName("Hege");

familyName("Stale");

familyName("Kai Jim");

familyName("Borge");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

function familyName($fname, $year) {

echo "$fname Refsnes. Born in $year <br>";

}

familyName("Hege","1975");

familyName("Stale","1978");

familyName("Kai Jim","1983");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

function setHeight($minheight = 50) {

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

}

setHeight(350);

setHeight();

setHeight(135);

setHeight(80);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

function sum($x, $y) {

$z = $x + $y;

return $z;

}

echo "5 + 10 = " . sum(5,10) . "<br>";

echo "7 + 13 = " . sum(7,13) . "<br>";

echo "2 + 4 = " . sum(2,4);

?>

</body>

</html>

# Arrays

An array stores multiple values in one single variable:

<!DOCTYPE html>

<html>

<body>

<?php

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

echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";

?>

</body>

</html>

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

Indexed Arrays

There are two ways to create indexed arrays:

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

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

or the index can be assigned manually:

$cars[0] = "Volvo";  
$cars[1] = "BMW";  
$cars[2] = "Toyota";

<!DOCTYPE html>

<html>

<body>

<?php

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

echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

echo count($cars);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

$arrlength = count($cars);

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

echo $cars[$x];

echo "<br>";

}

?>

</body>

</html>

Associative Arrays

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

There are two ways to create an associative array:

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

or:

$age['Peter'] = "35";  
$age['Ben'] = "37";  
$age['Joe'] = "43";

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

echo "Peter is " . $age['Peter'] . " years old.";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x\_value) {

echo "Key=" . $x . ", Value=" . $x\_value;

echo "<br>";

}

?>

</body>

</html>

Multidimensional Arrays

<!DOCTYPE html>

<html>

<body>

<?php

$cars = array

(

array("Volvo",22,18),

array("BMW",15,13),

array("Saab",5,2),

array("Land Rover",17,15)

);

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";

echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";

echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";

echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$cars = array

(

array("Volvo",22,18),

array("BMW",15,13),

array("Saab",5,2),

array("Land Rover",17,15)

);

for ($row = 0; $row < 4; $row++) {

echo "<p><b>Row number $row</b></p>";

echo "<ul>";

for ($col = 0; $col < 3; $col++) {

echo "<li>".$cars[$row][$col]."</li>";

}

echo "</ul>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

sort($cars);

$clength = count($cars);

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

echo $cars[$x];

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$numbers = array(4, 6, 2, 22, 11);

sort($numbers);

$arrlength = count($numbers);

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

echo $numbers[$x];

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

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

rsort($cars);

$clength = count($cars);

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

echo $cars[$x];

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$numbers = array(4, 6, 2, 22, 11);

rsort($numbers);

$arrlength = count($numbers);

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

echo $numbers[$x];

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

asort($age);

foreach($age as $x => $x\_value) {

echo "Key=" . $x . ", Value=" . $x\_value;

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

ksort($age);

foreach($age as $x => $x\_value) {

echo "Key=" . $x . ", Value=" . $x\_value;

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

arsort($age);

foreach($age as $x => $x\_value) {

echo "Key=" . $x . ", Value=" . $x\_value;

echo "<br>";

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

krsort($age);

foreach($age as $x => $x\_value) {

echo "Key=" . $x . ", Value=" . $x\_value;

echo "<br>";

}

?>

</body>

</html>

# Superglobals

Superglobals were introduced in PHP 4.1.0, and are built-in variables that are always available in all scopes.

Several predefined variables in PHP 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.

The PHP superglobal variables are:

* $GLOBALS
* $\_SERVER
* $\_REQUEST
* $\_POST
* $\_GET
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

<!DOCTYPE html>

<html>

<body>

<?php

$x = 75;

$y = 25;

function addition() {

$GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

}

addition();

echo $z;

?>

</body>

</html>

$\_SERVER['PHP\_SELF'] Returns the filename of the currently executing script

$\_SERVER['SERVER\_NAME'] Returns the name of the host server (such as [www.google.com](http://www.google.com))

$\_SERVER['HTTP\_HOST'] Returns the Host header from the current request

$\_SERVER['HTTP\_REFERER'] Returns the complete URL of the current page (not reliable because not all user-agents support it)

$\_SERVER['SCRIPT\_NAME'] Returns the path of the current script

<!DOCTYPE html>

<html>

<body>

<?php

echo $\_SERVER['PHP\_SELF'];

echo "<br>";

echo $\_SERVER['SERVER\_NAME'];

echo "<br>";

echo $\_SERVER['HTTP\_HOST'];

echo "<br>";

echo $\_SERVER['HTTP\_REFERER'];

echo "<br>";

echo $\_SERVER['HTTP\_USER\_AGENT'];

echo "<br>";

echo $\_SERVER['SCRIPT\_NAME'];

?>

</body>

</html>

PHP $\_REQUEST

PHP $\_REQUEST is used to collect data after submitting an HTML form.

PHP $\_POST

PHP $\_POST is widely used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.

<!DOCTYPE html>

<html>

<body>

<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">

Name: <input type="text" name="fname">

<input type="submit">

</form>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// collect value of input field

$name = htmlspecialchars($\_REQUEST['fname']);

if (empty($name)) {

echo "Name is empty";

} else {

echo $name;

}

}

?>

</body>

</html>

# Form

<!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>

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

<!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>

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

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.

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.

<!DOCTYPE html>

<html>

<body>

<?php

$str = addslashes('What does "yolo" mean?');// The addslashes() function returns a string with backslashes in front of predefined characters.

echo($str);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

echo stripslashes("Who\'s Peter Griffin?");// The stripslashes() function removes backslashes added by the addslashes() function.

?>

</body>

</html>

PHP 5 Form Validation

<!DOCTYPE HTML>

<html>

<head>

</head>

<body>

<?php

// define variables and set to empty values

$name = $email = $gender = $comment = $website = "";

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$name = test\_input($\_POST["name"]);

$email = test\_input($\_POST["email"]);

$website = test\_input($\_POST["website"]);

$comment = test\_input($\_POST["comment"]);

$gender = test\_input($\_POST["gender"]);

}

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<h2>PHP Form Validation Example</h2>

<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

Name: <input type="text" name="name">

<br><br>

E-mail: <input type="text" name="email">

<br><br>

Website: <input type="text" name="website">

<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

<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>

PHP - Required Fields

<!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"]);

}

if (empty($\_POST["email"])) {

$emailErr = "Email is required";

} else {

$email = test\_input($\_POST["email"]);

}

if (empty($\_POST["website"])) {

$website = "";

} else {

$website = test\_input($\_POST["website"]);

}

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

<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>

<!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

<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>

# PHP 5 Date and Time

* d - Represents the day of the month (01 to 31)
* m - Represents a month (01 to 12)
* Y - Represents a year (in four digits)
* l (lowercase 'L') - Represents the day of the week
* h - 12-hour format of an hour with leading zeros (01 to 12)
* i - Minutes with leading zeros (00 to 59)
* s - Seconds with leading zeros (00 to 59)
* a - Lowercase Ante meridiem and Post meridiem (am or pm)

<!DOCTYPE html>

<html>

<body>

<?php

echo "Today is " . date("Y/m/d") . "<br>";

echo "Today is " . date("Y.m.d") . "<br>";

echo "Today is " . date("Y-m-d") . "<br>";

echo "Today is " . date("l");

&copy; 2010-<?php echo date("Y");?>

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

echo "The time is " . date("h:i:sa");

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

date\_default\_timezone\_set("America/New\_York");

echo "The time is " . date("h:i:sa");

?>

</body>

</html>

mktime()

Syntax

mktime(hour,minute,second,month,day,year)

The mktime() function returns the Unix timestamp for a date. The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

<!DOCTYPE html>

<html>

<body>

<?php

$d=mktime(11, 14, 54, 8, 12, 2014);

echo "Created date is " . date("Y-m-d h:i:sa", $d);

?>

</body>

</html>

## PHP strtotime()

The PHP strtotime() function is used to convert a human readable string to a Unix time.

### Syntax

strtotime(time,now)

<!DOCTYPE html>

<html>

<body>

<?php

$d=strtotime("10:30pm April 15 2014");

echo "Created date is " . date("Y-m-d h:i:sa", $d);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$d=strtotime("tomorrow");

echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("next Saturday");

echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("+3 Months");

echo date("Y-m-d h:i:sa", $d) . "<br>";

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$startdate=strtotime("Saturday");

$enddate=strtotime("+6 weeks", $startdate);

while ($startdate < $enddate) {

echo date("M d", $startdate) . "<br>";

$startdate = strtotime("+1 week", $startdate);

}

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

var\_dump(checkdate(12,31,-400));

echo "<br>";

var\_dump(checkdate(2,29,2004));

$date=date\_create("2013-03-15");

date\_add($date,date\_interval\_create\_from\_date\_string("40 days"));

echo date\_format($date,"Y-m-d");

$date=date\_create("2013-03-15");

echo date\_format($date,"Y/m/d");

$date=date\_create();

date\_date\_set($date,2020,10,30);

echo date\_format($date,"Y/m/d");

echo date\_default\_timezone\_get();

$date=date\_create("2013-05-01");

date\_modify($date,"+15 days");

echo date\_format($date,"Y-m-d");

$date=date\_create("2013-03-15");

date\_sub($date,date\_interval\_create\_from\_date\_string("40 days"));

echo date\_format($date,"Y-m-d");

// Print the array from getdate()

print\_r(getdate());

echo "<br><br>";

// Return date/time info of a timestamp; then format the output

$mydate=getdate(date("U"));

echo "$mydate[weekday], $mydate[month] $mydate[mday], $mydate[year]";

?>

</body>

</html>

# Include Files

The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

<!DOCTYPE html>

<html>

<body>

<div class="menu">

<?php include 'menu.php';?>

</div>

<h1>Welcome to my home page!</h1>

<p>Some text.</p>

<p>Some more text.</p>

</body>

</html>

file is included with the **include** statement and PHP cannot find it, the script will continue to execute:

<!DOCTYPE html>

<html>

<body>

<h1>Welcome to my home page!</h1>

<?php include 'noFileExists.php';

echo "I have a $color $car.";

?>

</body>

</html>

the **require** statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

<!DOCTYPE html>

<html>

<body>

<h1>Welcome to my home page!</h1>

<?php require 'noFileExists.php';

echo "I have a $color $car.";

?>

</body>

</html>

# File Handling

readfile() Function: The readfile() function is useful if all you want to do is open up a file and read its contents.

<!DOCTYPE html>

<html>

<body>

<?php

echo readfile("webdictionary.txt");

?>

</body>

</html>

fopen()

<!DOCTYPE html>

<html>

<body>

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

echo fread($myfile,filesize("webdictionary.txt"));

fclose($myfile);

?>

</body>

</html>

The first parameter of fopen() contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

fread()

The fread() function reads from an open file.

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

|  |  |
| --- | --- |
| **Modes** | **Description** |
| r | **Open a file for read only**. File pointer starts at the beginning of the file |
| w | **Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a | **Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x | **Creates a new file for write only**. Returns FALSE and an error if file already exists |
| r+ | **Open a file for read/write**. File pointer starts at the beginning of the file |
| w+ | **Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file |
| a+ | **Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist |
| x+ | **Creates a new file for read/write**. Returns FALSE and an error if file already exists |

fgets()

The fgets() function is used to read a single line from a file.

<!DOCTYPE html>

<html>

<body>

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

echo fgets($myfile);

fclose($myfile);

?>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

// Output one line until end-of-file

while(!feof($myfile)) {

echo fgets($myfile) . "<br>";

}

fclose($myfile);

?>

</body>

</html>

fgetc()

The fgetc() function is used to read a single character from a file.

<!DOCTYPE html>

<html>

<body>

<?php

$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");

// Output one character until end-of-file

while(!feof($myfile)) {

echo fgetc($myfile);

}

fclose($myfile);

?>

</body>

</html>

fwrite()

The fwrite() function is used to write to a file.

The first parameter of fwrite() contains the name of the file to write to and the second parameter is the string to be written.

<?php

$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");

$txt = "John Doe\n";

fwrite($myfile, $txt);

$txt = "Jane Doe\n";

fwrite($myfile, $txt);

fclose($myfile);

?>

# PHP 5 File Upload

$\_FILES:The PHP global $\_FILES contains all the information of file. By the help of $\_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is filename.

$\_FILES['filename']['name']:returns file name.

$\_FILES['filename']['type']:returns MIME type of the file.

$\_FILES['filename']['size']:returns size of the file (in bytes).

$\_FILES['filename']['tmp\_name']:returns temporary file name of the file which was stored on the server.

$\_FILES['filename']['error']:returns error code associated with this file.

move\_uploaded\_file() function

The move\_uploaded\_file() function moves the uploaded file to a new location. The move\_uploaded\_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

bool move\_uploaded\_file ( string $filename , string $destination )

<!DOCTYPE html>

<html>

<body>

<form action="upload.php" method="post" enctype="multipart/form-data">

Select image to upload:

<input type="file" name="fileToUpload" id="fileToUpload">

<input type="submit" value="Upload Image" name="submit">

</form>

</body>

</html>

<?php

$target\_dir = "uploads/";

$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);

$uploadOk = 1;

$imageFileType = pathinfo($target\_file,PATHINFO\_EXTENSION);

// Check if image file is a actual image or fake image

if(isset($\_POST["submit"])) {

$check = getimagesize($\_FILES["fileToUpload"]["tmp\_name"]);

if($check !== false) {

echo "File is an image - " . $check["mime"] . ".";

$uploadOk = 1;

} else {

echo "File is not an image.";

$uploadOk = 0;

}

}

// Check if file already exists

if (file\_exists($target\_file)) {

echo "Sorry, file already exists.";

$uploadOk = 0;

}

// Check file size

if ($\_FILES["fileToUpload"]["size"] > 500000) {

echo "Sorry, your file is too large.";

$uploadOk = 0;

}

// Allow certain file formats

if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"

&& $imageFileType != "gif" ) {

echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";

$uploadOk = 0;

}

// Check if $uploadOk is set to 0 by an error

if ($uploadOk == 0) {

echo "Sorry, your file was not uploaded.";

// if everything is ok, try to upload file

} else {

if (move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file)) {

echo "The file ". basename( $\_FILES["fileToUpload"]["name"]). " has been uploaded.";

} else {

echo "Sorry, there was an error uploading your file.";

}

}

?>

PHP Download File Example: Text File

<?php

$file\_url = 'http://www.javatpoint.com/f.txt';

header('Content-Type: application/octet-stream');

header("Content-Transfer-Encoding: utf-8");

header("Content-disposition: attachment; filename=\"" . basename($file\_url) . "\"");

readfile($file\_url);

?>

# Cookies

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.

A cookie is created with the setcookie() function.

### Syntax

setcookie(name, value, expire, path, domain, secure, httponly);

<!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];

}

?>

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 5 Sessions

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.

PHP session is started with the session\_start() function.

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

Page1.php

<?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>

Page2.php

<?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>

Page3.php

<?php

session\_start();

?>

<!DOCTYPE html>

<html>

<body>

<?php

print\_r($\_SESSION);

?>

</body>

</html>

Modify a PHP Session Variable

Page4,php

<?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

<?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>

**Login Page**

<?php

session\_start();

?>

<html lang = "en">

<head>

<title>abc.com</title>

<link href = "css/bootstrap.min.css" rel = "stylesheet">

<style>

body {

padding-top: 40px;

padding-bottom: 40px;

background-color: #ADABAB;

}

.form-signin {

max-width: 330px;

padding: 15px;

margin: 0 auto;

color: #017572;

}

.form-signin .form-signin-heading,

.form-signin .checkbox {

margin-bottom: 10px;

}

.form-signin .checkbox {

font-weight: normal;

}

.form-signin .form-control {

position: relative;

height: auto;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

box-sizing: border-box;

padding: 10px;

font-size: 16px;

}

.form-signin .form-control:focus {

z-index: 2;

}

.form-signin input[type="email"] {

margin-bottom: -1px;

border-bottom-right-radius: 0;

border-bottom-left-radius: 0;

border-color:#017572;

}

.form-signin input[type="password"] {

margin-bottom: 10px;

border-top-left-radius: 0;

border-top-right-radius: 0;

border-color:#017572;

}

h2{

text-align: center;

color: #017572;

}

</style>

</head>

<body>

<h2>Enter Username and Password</h2>

<div class = "container form-signin">

<?php

$msg = '';

if (isset($\_POST['login']) && !empty($\_POST['username'])

&& !empty($\_POST['password'])) {

if ($\_POST['username'] == 'dipika' &&

$\_POST['password'] == '1234') {

$\_SESSION['valid'] = true;

$\_SESSION['timeout'] = time();

$\_SESSION['username'] = 'dipika';

echo 'You have entered valid use name and password';

}else {

$msg = 'Wrong username or password';

}

}

?>

</div> <!-- /container -->

<div class = "container">

<form class = "form-signin" role = "form"

action = "<?php echo htmlspecialchars($\_SERVER['PHP\_SELF']);

?>" method = "post">

<h4 class = "form-signin-heading"><?php echo $msg; ?></h4>

<input type = "text" class = "form-control"

name = "username" placeholder = "username = dipika"

required autofocus></br>

<input type = "password" class = "form-control"

name = "password" placeholder = "password = 1234" required>

<button class = "btn btn-lg btn-primary btn-block" type = "submit"

name = "login">Login</button>

</form>

Click here to clean <a href = "logout.php" tite = "Logout">Session.

</div>

</body>

</html>

Logout.php

<?php

session\_start();

unset($\_SESSION["username"]);

unset($\_SESSION["password"]);

echo 'You have cleaned session';

header('Refresh: 2; URL = login.php');

?>

# PHP Filters

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

border-collapse: collapse;

}

th, td {

padding: 5px;

}

</style>

</head>

<body>

<table>

<tr>

<td>Filter Name</td>

<td>Filter ID</td>

</tr>

<?php

foreach (filter\_list() as $id =>$filter) {

echo '<tr><td>' . $filter . '</td><td>' . filter\_id($filter) . '</td></tr>';

}

?>

</table>

## Error Handling

Error handling is the process of catching errors raised by your program and then taking appropriate action.

## die() function

<?php

if(!file\_exists("welcome.txt")) {

die("File not found");

} else {

$file=fopen("welcome.txt","r");

}

?>

## Custom Error Handler

<?php

//error handler function

function customError($errno, $errstr) {

echo "<b>Error:</b> [$errno] $errstr";

}

//set error handler

set\_error\_handler("customError");

//trigger error

echo($test);

?>

## Trigger an Error

## Eg:

<?php

$test=2;

if ($test>=1) {

trigger\_error("Value must be 1 or below");

}

?>

Eg:

<?php  
//error handler function  
function customError($errno, $errstr) {  
  echo "<b>Error:</b> [$errno] $errstr<br>";  
  echo "Ending Script";  
  die();  
}  
  
//set error handler  
set\_error\_handler("customError",E\_USER\_WARNING);  
  
//trigger error  
$test=2;  
if ($test>=1) {  
  trigger\_error("Value must be 1 or below",E\_USER\_WARNING);  
}  
?>

# Exception Handling

<?php

//create function with an exception

function checkNum($number) {

if($number>1) {

throw new Exception("Value must be 1 or below");

}

return true;

}

//trigger exception

checkNum(2);

?>

Eg:

<?php

//create function with an exception

function checkNum($number) {

if($number>1) {

throw new Exception("Value must be 1 or below");

}

return true;

}

//trigger exception in a "try" block

try {

checkNum(2);

//If the exception is thrown, this text will not be shown

echo 'If you see this, the number is 1 or below';

}

//catch exception

catch(Exception $e) {

echo 'Message: ' .$e->getMessage();

}

?>

**PHP and MySql**

PHP 5 and MySQL database using:

* **MySQLi extension** (the "i" stands for improved)
* 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();

?>

Eg:

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// sql to create table

$sql = "CREATE TABLE MyGuests (

id INT(6) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

firstname VARCHAR(30) NOT NULL,

lastname VARCHAR(30) NOT NULL,

email VARCHAR(50),

reg\_date TIMESTAMP

)";

if ($conn->query($sql) === TRUE) {

echo "Table MyGuests created successfully";

} else {

echo "Error creating table: " . $conn->error;

}

$conn->close();

?>

Eg:

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {

echo "New record created successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

?>

# Insert Multiple Records Into MySQL

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('John', 'Doe', 'john@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Mary', 'Moe', 'mary@example.com');";

$sql .= "INSERT INTO MyGuests (firstname, lastname, email)

VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi\_query($sql) === TRUE) {

echo "New records created successfully";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

$conn->close();

?>

# Prepared Statements

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// prepare and bind

$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $firstname, $lastname, $email);

// set parameters and execute

$firstname = "John";

$lastname = "Doe";

$email = "john@example.com";

$stmt->execute();

$firstname = "Mary";

$lastname = "Moe";

$email = "mary@example.com";

$stmt->execute();

$firstname = "Julie";

$lastname = "Dooley";

$email = "julie@example.com";

$stmt->execute();

echo "New records created successfully";

$stmt->close();

$conn->close();

?>

# Select Data From MySQL

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

// output data of each row

while($row = $result->fetch\_assoc()) {

echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";

}

} else {

echo "0 results";

}

$conn->close();

?>

Eg:

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td {

border: 1px solid black;

}

</style>

</head>

<body>

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "SELECT id, firstname, lastname FROM MyGuests";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

echo "<table><tr><th>ID</th><th>Name</th></tr>";

// output data of each row

while($row = $result->fetch\_assoc()) {

echo "<tr><td>" . $row["id"]. "</td><td>" . $row["firstname"]. " " . $row["lastname"]. "</td></tr>";

}

echo "</table>";

} else {

echo "0 results";

}

$conn->close();

?>

</body>

</html>

# Delete Data From MySQL

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// sql to delete a record

$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {

echo "Record deleted successfully";

} else {

echo "Error deleting record: " . $conn->error;

}

$conn->close();

?>

# Update Data in MySQL

<?php

$servername = "localhost";

$username = "username";

$password = "password";

$dbname = "myDB";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if ($conn->query($sql) === TRUE) {

echo "Record updated successfully";

} else {

echo "Error updating record: " . $conn->error;

}

$conn->close();

?>

PHP XML

note.xml

<?xml version="1.0" encoding="UTF-8"?>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>

<?php  
$xmlDoc = new DOMDocument();  
$xmlDoc->load("note.xml");  
  
$x = $xmlDoc->documentElement;  
foreach ($x->childNodes AS $item) {  
  print $item->nodeName . " = " . $item->nodeValue . "<br>";  
}  
?>

Database

getuser.php

<!DOCTYPE html>  
<html>  
<head>  
<style>  
table {  
    width: 100%;  
    border-collapse: collapse;  
}  
  
table, td, th {  
    border: 1px solid black;  
    padding: 5px;  
}  
  
th {text-align: left;}  
</style>  
</head>  
<body>  
  
<?php  
$q = intval($\_GET['q']);  
  
$con = mysqli\_connect('localhost','peter','abc123','my\_db');  
if (!$con) {  
    die('Could not connect: ' . mysqli\_error($con));  
}  
  
mysqli\_select\_db($con,"ajax\_demo");  
$sql="SELECT \* FROM user WHERE id = '".$q."'";  
$result = mysqli\_query($con,$sql);  
  
echo "<table>  
<tr>  
<th>Firstname</th>  
<th>Lastname</th>  
<th>Age</th>  
<th>Hometown</th>  
<th>Job</th>  
</tr>";  
while($row = mysqli\_fetch\_array($result)) {  
    echo "<tr>";  
    echo "<td>" . $row['FirstName'] . "</td>";  
    echo "<td>" . $row['LastName'] . "</td>";  
    echo "<td>" . $row['Age'] . "</td>";  
    echo "<td>" . $row['Hometown'] . "</td>";  
    echo "<td>" . $row['Job'] . "</td>";  
    echo "</tr>";  
}  
echo "</table>";  
mysqli\_close($con);  
?>  
</body>  
</html>

<!DOCTYPE html>

<html>

<head>

<script>

function showUser(str) {

if (str=="") {

document.getElementById("txtHint").innerHTML="";

return;

}

if (window.XMLHttpRequest) {

// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

} else { // code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function() {

if (this.readyState==4 && this.status==200) {

document.getElementById("txtHint").innerHTML=this.responseText;

}

}

xmlhttp.open("GET","getuser.php?q="+str,true);

xmlhttp.send();

}

</script>

</head>

<body>

<form>

<select name="users" onchange="showUser(this.value)">

<option value="">Select a person:</option>

<option value="1">Peter Griffin</option>

<option value="2">Lois Griffin</option>

<option value="3">Joseph Swanson</option>

<option value="4">Glenn Quagmire</option>

</select>

</form>

<br>

<div id="txtHint"><b>Person info will be listed here.</b></div>

</body>

</html>

## mail() function

<?php

$to = "abc@example.com";//change receiver address

$subject = "This is subject";

$message = "<h1>This is HTML heading</h1>";

$header = "From:xyz@example.com \r\n";

$header .= "MIME-Version: 1.0 \r\n";

$header .= "Content-type: text/html;charset=UTF-8 \r\n";

$result = mail ($to,$subject,$message,$header);

if( $result == true ){

echo "Message sent successfully...";

}else{

echo "Sorry, unable to send mail...";

}

?>

Eg:

<?php

$to = "abc@example.com";

$subject = "This is subject";

$message = "This is a text message.";

# Open a file

$file = fopen("/tmp/test.txt", "r" );//change your file location

if( $file == false )

{

echo "Error in opening file";

exit();

}

$size = filesize("/tmp/test.txt");

$content = fread( $file, $size);

//encode the data for safe transit

// and insert \r\n after every 76 chars.

$encoded\_content = chunk\_split( base64\_encode($content));

// Get a random 32 bit number using time() as seed.

$num = md5( time() );

//Define the main headers.

$header = "From:xyz@example.com\r\n";

$header .= "MIME-Version: 1.0\r\n";

$header .= "Content-Type: multipart/mixed; ";

$header .= "boundary=$num\r\n";

$header .= "--$num\r\n";

//Define the message section

$header .= "Content-Type: text/plain\r\n";

$header .= "Content-Transfer-Encoding:8bit\r\n\n";

$header .= "$message\r\n";

$header .= "--$num\r\n";

//Define the attachment section

$header .= "Content-Type: multipart/mixed; ";

$header .= "name=\"test.txt\"\r\n";

$header .= "Content-Transfer-Encoding:base64\r\n";

$header .= "Content-Disposition:attachment; ";

$header .= "filename=\"test.txt\"\r\n\n";

$header .= "$encoded\_content\r\n";

$header .= "--$num--";

//Send email now

$result = mail ( $to, $subject, "", $header );

if( $result == true ){

echo "Message sent successfully...";

}else{

echo "Sorry, unable to send mail...";

}

?>