**What is PHP?**

=>PHP is a server side scripting language.

=>that is used to develop Static websites or Dynamic websites or Web applications.

=>PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.

=>PHP scripts can only be interpreted on a server that has PHP installed.

=>A PHP file contains PHP tags and ends with the extension “.php”.

## PHP Data Types

* Alphanumeric characters are classified as strings
* Whole numbers are classified integers
* Numbers with decimal points are classified as floating points.
* True or false values are classified as Boolean.

Integer – whole numbers e.g. -3, 0, 69. The maximum value of an integer is platform-dependent. On a 32 bit machine, it’s usually around 2 billion. 64 bit machines usually have larger values. The constant PHP\_INT\_MAX is used to determine the maximum value.

Floating point number – decimal numbers e.g. 3.14. they are also known as double or real numbers. The maximum value of a float is platform-dependent. Floating point numbers are larger than integers.

Character string – e.g. Hello World

Boolean – e.g. True or false.

## PHP Constant

**Define constant**– A constant is a variable whose value cannot be changed at runtime.

Suppose we are developing a program that uses the value of PI 3.14, we can use a constant to store its value.

## PHP Operators

### ***Arithmetic operators***

Arithmetic operators are used to perform arithmetic operations on numeric data. The concatenate operator works on strings values too. PHP supports the following operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| + | Addition | Summation of x and y | 1 + 1; | 2 |
| – | Subtraction | Difference between x and y | 1 – 1; | 0 |
| \* | Multiplication | Multiplies x and y | 3 \* 7; | 21 |
| / | Division | Quotient of x and y | 45 / 5; | 9 |
| % | PHP Modulus | Gives remainder of dividing x and y | 10 % 3; | 1 |
| -n | Negation | Turns n into a negative number | -(-5); | 5 |
| x . y | Concatenation | Puts together x and y | “PHP” . ” ROCKS”;10 . 3; | PHP ROCKS103 |

### ***Assignment Operators***

Assignment operators are used to assign values to variables. They can also be used together with arithmetic operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| x = ? | assignment | Assigns the value of x to ? | $x = 5; | 5 |
| x += ? | addition | Increments the value of x by ? | $x = 2;$x += 1; | 3 |
| X -= ? | subtraction | Subtracts ? from the value of x | $x = 3;$x -= 2; | 1 |
| X \*=? | multiplication | Multiplies the value of x ? times | $x = 0;$x \*=9; | 0 |
| X /=? | division | Quotient of x and ? | $x = 6;$x /=3; | 2 |
| X %=? | modulus | The reminder of dividing x by? | $x = 3;$x %= 2; | 1 |
| X .=? | concatenate | Puts together items | ” $x = ‘Pretty’;$x .= ‘ Cool!’;” | Pretty Cool! |

### ***Comparison operators***

Comparison operators are used to compare values and data types.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X == y | Equal | Compares x and y then returns true if they are equal | 1 == “1”; | True or 1 |
| X === y | identical | Compares both values and data types. | 1 === “1”; | False or 0. Since 1 is integer and “1” is string |
| X != y, x <> y | PHP Not equal | Compares values of x and y. returns true if the values are not equal | 2 != 1; | True or 1 |
| X > y | Greater than | Compares values of x and y. returns true if x is greater than y | 3 > 1; | True or 1 |
| X < y | Less than | Compares values of x and y. returns true if x is less than y | 2 < 1; | False or 0 |
| X >= y | Greater than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 1 >=1 | True or 1 |
| X <= y | Less than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 8 <= 6 | False or 0 |

### ***Logical operators***

When working with logical operators, any number greater than or less than zero (0) evaluates to true. Zero (0) evaluates to false.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X and y, x && y | And | Returns true if both x and y are equal | 1 and 4;True&& False; | True or 1False or 0 |
| X or y, x || y | Or | Returns true if either x or y is true | 6 or 9;0 || 0; | True or 1False or 0 |
| X xor y | Exclusive or, xor | Returns true if only x is true or only y is true | 1 xor 1;1 xor 0; | False or 0True or 1 |
| !x | Not | Returns true if x is false and false if x is true | !0; | True or 1 |

## PHP Comments

:-

[PHP Include, Require & Comments](https://www.guru99.com/images/2013/04/php_single_comment.jpg)single line comment

/\* any thinkkkkk \*/ multiline comment….

## What is a PHP Array?

A PHP array is a variable that stores more than one piece of related data in a single variable.

* [Numeric Arrays](https://www.guru99.com/arrays.html#1):-

$movie[0] = 'Shaolin Monk';

* [PHP Associative Array](https://www.guru99.com/arrays.html#2):-

$variable\_name['key\_name'] = value;

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

* [PHP Multi-dimensional arrays](https://www.guru99.com/arrays.html#3):-

$movies =array(

"comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),

"action" => array("Die Hard", "Expendables"),

"epic" => array("The Lord of the rings"),

"Romance" => array("Romeo and Juliet")

);

* [PHP Array operators](https://www.guru99.com/arrays.html#4):-

|  |  |
| --- | --- |
| x + y | Union |
| X == y | Equal |
| X === y | Identical |
| X != y, x <> y | Not equal |
| X !== y | Non identical |

## PHP Array Functions

### Count function

The count function is used to count the number of elements that an php array contains. The code below shows the implementation.

<?php

$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");

echo count($lecturers);

?>

### is\_array function

The is\_array function is used to determine if a variable is an array or not. Let’s now look at an example that implements the is\_array functions.

<?php

$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");

echo is\_array($lecturers);

?>

### Sort

This function is used to sort arrays by the values.

If the values are alphanumeric, it sorts them in alphabetical order.

If the values are numeric, it sorts them in ascending order.

It removes the existing access keys and add new numeric keys.

The output of this function is a numeric array

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

sort($persons);

print\_r($persons);

?>

### ksort

This function is used to sort the array using the key. The following example illustrates its usage.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

ksort($persons);

print\_r($persons);

?>

### asort

This function is used to sort the array using the values. The following example illustrates its usage.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

asort($persons);

print\_r($persons);

?>

**Let's introduce some Logic!**

## What is a control structure?

Code execution can be grouped into categories as shown below

* **Sequential**– this one involves executing all the codes in the order in which they have been written.
* **Decision** – this one involves making a choice given a number of options. The code executed depends on the value of the condition.

## PHP IF Else

If… then… else is the **simplest** **control** **structure**. It evaluates the conditions using Boolean logic  
**When to use if… then… else**

**Syntax** The syntax for if… then… else is;

<?php

if (condition is true) {

block one

else

block two

}

?>

Example:-

<?php

$first\_number = 7;

$second\_number = 21;

if ($first\_number > $second\_number){

echo "$first\_number is greater than $second\_number";

}else{

echo "$second\_number is greater than $first\_number";

}

?>

## PHP Switch Case

**Switch… case** is similar to the **if then… else** control structure.

It only **executes** a single block of code depending on the **value** of the condition.

If no condition has been met then the default block of code is executed.

It has the following basic syntax.

<?php

switch(condition){

case value:

//block of code to be executed

break;

case value2:

//block of code to be executed

break;

default:

//default block code

break;

}

?>

Example:-

<?php

$today = "wednesday";

switch($today){

case "sunday":

echo "pray for us sinners.";

break;

case "wednesday":

echo "ladies night, take her out for dinner";

break;

case "saturday":

echo "take care as you go out tonight.";

break;

default:

echo "have a nice day at work";

break;

}

?>

# PHP Loop

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

### PHP For Loop

<?php

for (initialize; condition; increment){

//code to be executed

}

?>

Example:-

<?php

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

$product = 10 \* $i;

echo "The product of 10 \* $i is $product <br/>";

}

?>

### PHP For Each loop

The php foreach loop is used to iterate through array values. It has the following basic syntax

<?php

foreach($array\_variable as $array\_values){

block of code to be executed

}

?>

**Practical examples**

The code below uses for… each loop to read and print the elements of an array.

<?php

$animals\_list = array("Lion","Wolf","Dog","Leopard","Tiger");

foreach($animals\_list as $array\_values){

echo $array\_values . "<br>";

}

?>

Both for loop and foreach loop are used to repeat a set of statements, but the syntax is different. The key difference between for Loop and foreach loop is that **the for loop is a general purpose control structure while the foreach loop is an enhanced for loop that is applicable only to arrays and collections**.

## While Loop

### PHP While loop

They are used to execute a block of code a repeatedly until the set condition gets satisfied

**Types of while loops**

* **Do… while** – executes the block of code at least once before evaluating the condition
* **While…** – checks the condition first. If it evaluates to true, the block of code is executed as long as the condition is true. If it evaluates to false, the execution of the while loop is terminated.

**While loop**

It has the following syntax

<?php

while (condition){

block of code to be executed;

}

?>

**Practical example**

The code below uses the while… loop to print numbers 1 to 5.

<?php

$i = 0;

while ($i < 5){

echo $i + 1 . "<br>";

$i++;

}

?>

### PHP Do While

The difference between While… loop and Do… while loop is do… while is executed at-least once before the condition is evaluated.

<?php

do{

block of code to be executed

}

?>

Example:-

<?php

$i = 9;

do{

echo "$i is"." <br>";

}

while($i < 9);

?>

## What is String in PHP?

A string is a collection of characters. String is one of the data types supported by PHP.

**PHP Create Strings Using Single quotes with Example**

<?php

var\_dump('You need to be logged in to view this page');

?>

**PHP Create Strings Using Double quotes with Example**

<?php

$name='Alicia';

echo "$name is friends with kalinda";

?>

**PHP Heredoc with Example**

This heredoc methodology is used to create fairly complex strings as compared to double quotes.

The heredoc supports all the features of double quotes and allows creating string values with more than one line without PHP string concatenation.

Using double quotes to create strings that have multiple lines generates an error.

You can also use double quotes inside without escaping them.

The example below illustrates how the Heredoc method is used to create string values.

<?php

$baby\_name = "Shalon";

echo <<<EOT

When $baby\_name was a baby,

She used to look like a "boy".

EOT;

?>

HERE,

**<<<EOT** is the string delimiter.

EOT is the acronym for end of text.

It should be defined in its on line at the beginning of the string and at the end.

**PHP Nowdoc with Example**

The Nowdoc string creation method is similar to the heredoc method but works like the way single quotes work.

No parsing takes place inside the Nowdoc.

Nowdoc is ideal when working with raw data that do not need to be parsed.

The code below shows the Nowdoc implementation

<?php

$baby\_name = "Shalon";

$my\_variable = <<<'EOT'

When $baby\_name was a baby,

She used to look like a "boy".

EOT;

echo $my\_variable;

?>

**PHP String Function Examples**

String functions in PHP are used to manipulate string values.

We are now going to look at some of the commonly used string functions in PHP

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| strtolower | Used to convert all string characters to lower case letters | echo strtolower( ‘Benjamin’); | outputs benjamin |
| strtoupper | Used to convert all string characters to upper case letters | echo strtoupper(‘george w bush’); | outputs GEORGE W BUSH |
| strlen | The string length function is used to count the number of character in a string. Spaces in between characters are also counted | echo strlen(‘united states of america’); | 24 |
| explode | Used to convert strings into an array variable | $settings = explode(‘;’, “host=localhost; db=sales; uid=root; pwd=demo”); print\_r($settings); | Array ( [0] => host=localhost [1] => db=sales [2] => uid=root [3] => pwd=demo ) |
| substr | Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned. | $my\_var = ‘This is a really long sentence that I wish to cut short’;echo substr($my\_var,0, 12).’…’; | This is a re… |
| str\_replace | Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is the replacement text and the third argument is the text that is analyzed. | echo str\_replace (‘the’, ‘that’, ‘the laptop is very expensive’); | that laptop is very expensive |
| strpos | Used to locate the and return the position of a character(s) within a string. This function accepts two arguments | echo strpos(‘PHP Programing’,’Pro’); | 4 |
| sha1 | Used to calculate the SHA-1 hash of a string value | echo sha1(‘password’); | 5baa61e4c 9b93f3f0 682250b6cf8331b 7ee68fd8 |
| md5 | Used to calculate the md5 hash of a string value | echo md5(‘password’); | 9f961034ee 4de758 baf4de09ceeb1a75 |
| str\_word\_count | Used to count the number of words in a string. | echo str\_word\_count (‘This is a really long sentence that I wish to cut short’); | 12 |
| ucfirst | Make the first character of a string value upper case | echo ucfirst(‘respect’); | Outputs Respect |
| lcfirst | Make the first character of a string value lower case | echo lcfirst(‘RESPECT’); | Outputs rESPECT |

For a complete list of PHP strings, check [https://php.net/manual/en/ref.strings.php](https://www.php.net/manual/en/ref.strings.php)

## strrev() - Reverse a String

The PHP strrev() function reverses a string.

### **Example**

Reverse the string "Hello world!":

<?php  
echo strrev("Hello world!"); // outputs !dlrow olleH  
?>

## PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

* [is\_nan()](https://www.w3schools.com/php/func_math_is_nan.asp)

## PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), or intval() function are often used to convert a value to an integer.

### **Example**

Cast float and string to integer:

<?php  
// Cast float to int  
$x = 23465.768;  
$int\_cast = (int)$x;  
echo $int\_cast;  
  
echo "<br>";  
  
// Cast string to int  
$x = "23465.768";  
$int\_cast = (int)$x;  
echo $int\_cast;  
?>

## 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:

### **Example**

<?php  
echo(min(0, 150, 30, 20, -8, -200));  // returns -200  
echo(max(0, 150, 30, 20, -8, -200));  // returns 150  
?>

## 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*)

<?php  
define("GREETING", "Welcome to W3Schools.com!");  
echo GREETING;  
?>

## PHP Constant Arrays

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

### **Example**

Create an Array constant:

<?php  
define("cars", [  
  "Alfa Romeo",  
  "BMW",  
  "Toyota"  
]);  
echo cars[0];  
?>

* sort() - sort arrays in ascending order
* rsort() - sort arrays in descending order
* asort() - sort associative arrays in ascending order, according to the value
* ksort() - sort associative arrays in ascending order, according to the key
* arsort() - sort associative arrays in descending order, according to the value
* krsort() - sort associative arrays in descending order, according to the key

## PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The index holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

### **Example**

<?php  
$x = 75;  
$y = 25;  
   
function addition() {  
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];  
}  
   
addition();  
echo $z;  
?>

$\_SERVER['PHP\_SELF']

=>Returns the filename of the currently executing script

$\_SERVER['GATEWAY\_INTERFACE']

=>Returns the version of the Common Gateway Interface (CGI) the server is using

$\_SERVER['SERVER\_ADDR']

=>  
Returns the IP address of the host server

$\_SERVER['SERVER\_NAME']

=>Returns the name of the host server (such as [www.w3schools.com](http://www.w3schools.com))

$\_SERVER['SERVER\_SOFTWARE']

=>Returns the server identification string (such as Apache/2.2.24)

$\_SERVER['SERVER\_PROTOCOL']

=>Returns the name and revision of the information protocol

$\_SERVER['REQUEST\_METHOD']

=>Returns the request method used to access the page (such as POST)

$\_SERVER['REQUEST\_TIME']

=>Returns the timestamp of the start of the request (such as 1377687496)

$\_SERVER['QUERY\_STRING']

=>Returns the query string if the page is accessed via a query string

$\_SERVER['HTTP\_ACCEPT']

=>  
Returns the Accept header from the current request

$\_SERVER['HTTP\_ACCEPT\_CHARSET']

=>Returns the Accept\_Charset header from the current request (such as utf-8,ISO-8859-1)

$\_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['HTTPS']

=>Is the script queried through a secure HTTP protocol

$\_SERVER['REMOTE\_ADDR']

=>Returns the IP address from where the user is viewing the current page

$\_SERVER['REMOTE\_HOST']

=>Returns the Host name from where the user is viewing the current page

$\_SERVER['REMOTE\_PORT']

=>Returns the port being used on the user's machine to communicate with the web server

$\_SERVER['SCRIPT\_FILENAME']

=>Returns the absolute pathname of the currently executing script

$\_SERVER['SERVER\_ADMIN']

=>Returns the value given to the SERVER\_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as [someone@w3schools.com](mailto:someone@w3schools.com))

$\_SERVER['SERVER\_PORT']

=>Returns the port on the server machine being used by the web server for communication (such as 80)

$\_SERVER['SERVER\_SIGNATURE']

=>Returns the server version and virtual host name which are added to server-generated pages

$\_SERVER['PATH\_TRANSLATED']

=>  
Returns the file system based path to the current script

$\_SERVER['SCRIPT\_NAME']

=>Returns the path of the current script

$\_SERVER['SCRIPT\_URI']

=>Returns the URI of the current page

## PHP $\_REQUEST

PHP $\_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_REQUEST to collect the value of the input field:

### **Example**

<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 = $\_REQUEST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>

## PHP $\_POST

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

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_POST to collect the value of the input field:

### **Example**

<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 = $\_POST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>

## PHP $\_GET

PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$\_GET can also collect data sent in the URL.

### **Example**

<html>  
<body>  
  
<?php  
echo "Study " . $\_GET['subject'] . " at " . $\_GET['web'];  
?>  
  
</body>  
</html>

## What is a Function in PHP?

A **Function in PHP** is a reusable piece or block of code that performs a specific action. It takes input from the user in the form of parameters, performs certain actions, and gives the output. Functions can either return values when called or can simply perform an operation without returning any value.

## Numeric Functions

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| is\_number | Accepts an argument and returns true if its numeric and false if it’s not | <?php  if(is\_numeric("guru"))  {  echo "true";  }  else  {  echo "false";  }  ?> | false |
| <?php  if(is\_numeric (123))  {  echo "true";  }  else  {  echo "false";  }  ?> | true |
| number\_format | Used to formats a numeric value using digit separators and decimal points | <?php  echo number\_format(2509663);  ?> | 2,509,663 |
| rand | Used to generate a random number. | <?php  echo rand();  ?> | Random number |
| round | Round off a number with decimal points to the nearest whole number. | <?php  echo round(3.49);  ?> | 3 |
| sqrt | Returns the square root of a number | <?php  echo sqrt(100);  ?> | 10 |
| cos | Returns the cosine | <?php  echo cos(45);  ?> | 0.52532198881773 |
| sin | Returns the sine | <?php  echo sin(45);  ?> | 0.85090352453412 |
| tan | Returns the tangent | <?php  echo tan(45);  ?> | 1.6197751905439 |
| pi | Constant that returns the value of PI | <?php  echo pi();  ?> | 3.1415926535898 |

**Date Function**

* Arrays – see the article on arrays for examples
* Files – see the article on files for examples
* Database functions – see the article on [MySQL PHP and other database access methods](https://www.guru99.com/mysql-php-and-other-database-access-methods.html) v2

**Why use User Defined Functions?**

* Function names must start with a letter or an underscore but not a number
* The function name must be unique
* The function name must not contain spaces
* It is considered a good practice to use descriptive function names.
* Functions can optionally accept parameters and return values too.

<?php

//define a function that displays hello function

function add\_numbers(){

echo 1 + 2;

}

add\_numbers ();

?>

## What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>…</form> tags and GUI items are defined using form elements such as input.

## Create a form

We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

* Opening and closing form tags <form>…</form>
* Form submission type POST or GET
* Submission URL that will process the submitted data
* Input fields such as input boxes, text areas, buttons,checkboxes etc.

**The code below creates a simple registration form**

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST"> First name:

<input type="text" name="firstname"> <br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

</body>

</html>

## GET vs POST Methods

**POST:-**

* Values not visible in the URL
* Has not limitation of the length of the values since they are submitted via the body of HTTP
* Has lower performance compared to Php\_GET method due to time spent encapsulation the Php\_POST values in the HTTP body
* Supports many different data types such as string, numeric, binary etc.
* Results cannot be book marked

**GET:-**

* Values visible in the URL
* Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
* Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
* Supports only string data types because the values are displayed in the URL
* Results can be book marked due to the visibility of the values in the URL

## Processing the registration form data

The registration form submits data to itself as specified in the action attribute of the form.

When a form has been submitted, the values are populated in the $\_POST super global array.

We will use the PHP isset function to check if the form values have been filled in the $\_POST array and process the data.

We will modify the registration form to include the PHP code that processes the data. Below is the modified code

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<?php if (isset($\_POST['form\_submitted'])): ?> //this code is executed when the form is submitted

<h2>Thank You <?php echo $\_POST['firstname']; ?> </h2>

<p>You have been registered as

<?php echo $\_POST['firstname'] . ' ' . $\_POST['lastname']; ?>

</p>

<p>Go <a href="/registration\_form.php">back</a> to the form</p>

<?php else: ?>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST">

First name:

<input type="text" name="firstname">

<br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

<?php endif; ? >

</body>

</html>

HERE,

* <?php if (isset($\_POST[‘form\_submitted’])): ?> checks if the form\_submitted hidden field has been filled in the $\_POST[] array and display a thank you and first name message.

If the form\_fobmitted field hasn’t been filled in the $\_POST[] array, the form is displayed.

<https://www.guru99.com/php-forms-handling.html> this link for more example….

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

Concate var with string..

### **Example**

<?php  
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;  
  
echo "<h2>" . $txt1 . "</h2>";  
echo "Study PHP at " . $txt2 . "<br>";  
echo $x + $y;  
?>

## What is Cookie?

A cookie is a small file with the maximum size of 4KB that the **web server stores on the client computer**.

Once a cookie has been set, all page requests that follow return the cookie name and value.

## Why and when to use Cookies?

* Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user’s computer.The path were the cookies are stored depends on the browser.Internet Explorer usually stores them in Temporal Internet Files folder.
* Personalizing the user experience – this is achieved by allowing users to select their preferences.The page requested that follow are personalized based on the set preferences in the cookies.
* Tracking the pages visited by a user

## Creating Cookies

Let’s now look at the basic syntax used to create a cookie.

<?php

setcookie(cookie\_name, cookie\_value, [expiry\_time], [cookie\_path], [domain], [secure], [httponly]);

?>

<?php

Setcookie(cookie\_name,cookie\_value,[expiry\_time],[cookie\_path],[domain],[secure],[httponly]);

?>

* Php“setcookie” is the PHP function used to create the cookie.

<?php

setcookie("user\_name", "Guru99", time()+ 60,'/'); // expires after 60 seconds

echo 'the cookie has been set for 60 seconds';

?>

**Output:**

the cookie has been set for 60 seconds

## Retrieving the Cookie value

Create another file named “cookies\_read.php” with the following code.

<?php

print\_r($\_COOKIE); //output the contents of the cookie array variable

?>

**Output:**

Array ( [PHPSESSID] => h5onbf7pctbr0t68adugdp2611 [user\_name] => Guru99 )

## What is a Session?

* A session is a global variable stored on the server.
* Each session is assigned a unique id which is used to retrieve stored values.
* Whenever a session is created, a cookie containing the unique session id is stored on the user’s computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
* Sessions have the capacity to store relatively large data compared to cookies.
* The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
* Just like the $\_COOKIE array variable, session variables are stored in the $\_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

## Creating a Session

In order to create a session, you must first call the PHP session\_start function and then store your values in the $\_SESSION array variable.

Let’s suppose we want to know the number of times that a page has been loaded, we can use a session to do that.

The code below shows how to create and retrieve values from sessions

<?php

session\_start(); //start the PHP\_session function

if(isset($\_SESSION['page\_count']))

{

$\_SESSION['page\_count'] += 1;

}

else

{

$\_SESSION['page\_count'] = 1;

}

echo 'You are visitor number ' . $\_SESSION['page\_count'];

?>

**Output:**

You are visitor number 1

## Destroying Session Variables

The session\_destroy() function is used to destroy the whole Php session variables.

If you want to destroy only a session single item, you use the unset() function.

The code below illustrates how to use both methods.

<?php

session\_destroy(); //destroy entire session

?>

<?php

unset($\_SESSION['product']); //destroy product session item

?>

Session\_destroy removes all the session data including cookies associated with the session.

Unset only frees the individual session variables.

Other data remains intact.

## What is a File?

A file is simply a resource for storing information on a computer.

Files are usually used to store information such as:

* Configuration settings of a program
* Simple data such as contact names against the phone numbers.
* Images, Pictures, Photos, etc.

[PHP File Formats Support](https://www.guru99.com/php-file-processing.html#1)

[PHP file() Function](https://www.guru99.com/php-file-processing.html#2)

[PHP file\_exists() Function](https://www.guru99.com/php-file-processing.html#3)

[PHP fopen() Function](https://www.guru99.com/php-file-processing.html#4)

[PHP fwrite() Function](https://www.guru99.com/php-file-processing.html#5)

[PHP fclose() Function](https://www.guru99.com/php-file-processing.html#6)

[PHP fgets() Function](https://www.guru99.com/php-file-processing.html#7)

[PHP copy() Function](https://www.guru99.com/php-file-processing.html#8)

[Deleting a file](https://www.guru99.com/php-file-processing.html#9)

[PHP file\_get\_contents() Function](https://www.guru99.com/php-file-processing.html#10)

## PHP File Formats Support

PHP file functions support a wide range of file formats that include:

* File.txt
* File.log
* File.custom\_extension i.e. file.xyz
* File.csv
* File.gif, file.jpg etc
* Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
* You want to store simple data such as server logs for later retrieval and analysis
* You want to store program settings i.e. program.ini

## PHP file\_exists() Function

This function is used to determine whether a file exists or not.

* It comes in handy when we want to know if a file exists or not before processing it.
* You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file\_exist function has the following syntax.

<?php

file\_exists($filename);

?>

<?php

if (file\_exists('my\_settings.txt'))

{

echo 'file found!';

}

else

{

echo 'my\_settings.txt does not exist';

}

?>

## PHP fopen() Function

The fopen function is used to open files. It has the following syntax

<?php

fopen($file\_name,$mode,$use\_include\_path,$context);

?>

|  |  |
| --- | --- |
| **Mode** | **Description** |
| r | * Read file from beginning. * Returns false if the file doesn’t exist. * Read only |
| r+ | * Read file from beginning * Returns false if the file doesn’t exist. * Read and write |
| w | * Write to file at beginning * truncate file to zero length * If the file doesn’t exist attempt to create it. * Write only |
| w+ | * Write to file at beginning, truncate file to zero length * If the file doesn’t exist attempt to create it. * Read and Write |
| a | * Append to file at end * If the file doesn’t exist attempt to create it. * Write only |
| a+ | * Php append to file at end * If the file doesn’t exist attempt to create it * Read and write |

## PHP fwrite() Function

The fwrite function is used to write files.

It has the following syntax

<?php

fwrite($handle, $string, $length);

?>

HERE,

* “fwrite” is the PHP function for writing to files
* “$handle” is the file pointer resource
* “$string” is the data to be written in the file.
* “$length” is optional, can be used to specify the maximum file length.

## PHP fclose() Function

The fclose() function is used to close a file in php which is already open

It has the following syntax.s

<?php

fclose($handle);

?>

# **PHP Date and Time**

## The PHP Date() Function

The PHP date() function formats a timestamp to a more readable date and time.

### **Syntax**

date(*format*,*timestamp*)

### **Example**

<?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");  
?>

### **Example**

<?php  
echo "The time is " . date("h:i:sa");  
?>

//for time….

### **Example**

<?php  
date\_default\_timezone\_set("America/New\_York");  
echo "The time is " . date("h:i:sa");  
?>

//you can change default timezone set…

## Create a Date With mktime()

### **Syntax**

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

### **Example**

<?php  
$d=mktime(11, 14, 54, 8, 12, 2014);  
echo "Created date is " . date("Y-m-d h:i:sa", $d);  
?>

### **Example**

<?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>";  
?>

//sort in time..

Get date :-

<?php

echo "The time in " . date\_default\_timezone\_get() . " is " . date("H:i:s");

date\_default\_timezone\_set("Asia/Calcutta");

echo "The time in " . date\_default\_timezone\_get() . " is " . date("H:i:s");

?>

## PHP Time parameters

//Returns the full date and time

<?php

echo date("r");

?>

//Returns whether the current time is am or pm, AM or PM respectively

<?php

echo date("a");

echo date("A");

?>

//  
Returns the hour without leading zeroes [1 to 12], [0 to 23] respectively

<?php

echo date("g");

echo date("G");

?>

//Returns the hour with leading zeros [01 to 12],[00 to 23] respectively

<?php

echo date("h");

echo date("H");

?>

//Returns the minutes/seconds with leading zeroes [00 to 59]

<?php

echo date("i");

echo date("s");

?>

## Day parameters:-

## //Returns the day of the month with leading zeroes [01 to 31]

<?php

echo date("d");

?>

## //Returns the day of the month without leading zeroes [1 to 31]

<?php

echo date("j");

?>

## //Returns the first 3 letters of the day name [Sub to Sat]

<?php

echo date("D");

?>

## //Returns day name of the week [Sunday to Saturday]

<?php

echo date("l");

?>

//Returns day of the week without leading zeroes [0 to 6] Sunday is represent by zero (0) through to Saturday represented by six (6)

<?php

echo date("w");

?>

//Returns the day of the year without leading spaces [0 through to 365]

<?php

echo date("z");

?>

## Month Parameters

//Returns the month number with leading zeroes [01 to 12]

<?php

echo date("m");

?>

//  
Returns the month number without leading zeroes [01 to 12]

<?php

echo date("n");

?>

//Returns the first 3 letters of the month name [Jan to Dec]

<?php

echo date("M");

?>

//Returns the month name [January to December]

<?php

echo date("F");

?>

//Returns the number of days in a month [28 to 31]

<?php

echo date("t");

?>

# **PHP 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.

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

**The include and require statements are identical, except upon failure:**

* require will produce a fatal error (E\_COMPILE\_ERROR) and stop the script
* include will only produce a warning (E\_WARNING) and the script will continue

### **Syntax**

include '*filename*';  
  
or  
  
require '*filename*';

### **Example**

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<p>Some text.</p>  
<p>Some more text.</p>  
<?php include 'footer.php';?>  
  
</body>  
</html>

## PHP include vs. require

The require statement is also used to include a file into the PHP code.

However, there is one big difference between include and require; when a file is included with the include statement and PHP cannot find it, the script will continue to execute:

### **Example**

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php include 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

If we do the same example using the require statement, the echo statement will not be executed because the script execution dies after the require statement returned a fatal error:

### **Example**

<html>  
<body>  
  
<h1>Welcome to my home page!</h1>  
<?php require 'noFileExists.php';  
echo "I have a $color $car.";  
?>  
  
</body>  
</html>

**Use require when the file is required by the application.**

**Use include when the file is not required and application should continue when file is not found**.

## PHP readfile() Function

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

### **Example**

<?php  
echo readfile("webdictionary.txt");  
?>

## PHP Write to File - 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.

The example below writes a couple of names into a new file called "newfile.txt":

### **Example**

<?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 Overwriting

Now that "newfile.txt" contains some data we can show what happens when we open an existing file for writing. All the existing data will be ERASED and we start with an empty file.

In the example below we open our existing file "newfile.txt", and write some new data into it:

### **Example**

<?php  
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");  
$txt = "Mickey Mouse\n";  
fwrite($myfile, $txt);  
$txt = "Minnie Mouse\n";  
fwrite($myfile, $txt);  
fclose($myfile);  
?>

If we now open the "newfile.txt" file, both John and Jane have vanished, and only the data we just wrote is present:

Mickey Mouse  
Minnie Mouse

## PHP Append Text

You can append data to a file by using the "a" mode. The "a" mode appends text to the end of the file, while the "w" mode overrides (and erases) the old content of the file.

In the example below we open our existing file "newfile.txt", and append some text to it:

### **Example**

<?php  
$myfile = fopen("newfile.txt", "a") or die("Unable to open file!");  
$txt = "Donald Duck\n";  
fwrite($myfile, $txt);  
$txt = "Goofy Goof\n";  
fwrite($myfile, $txt);  
fclose($myfile);  
?>

If we now open the "newfile.txt" file, we will see that Donald Duck and Goofy Goof is appended to the end of the file:

Mickey Mouse  
Minnie Mouse  
Donald Duck  
Goofy Goof

Practis…:-

<?php

    include 'sample.txt';

?>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h1>php in html</h1>

    <?php

        echo readfile("sample.txt");

    ?>

    <?php

        $r = fopen("sample.txt","w");

        $gns = "hello this is a fileopen to use...";

        fwrite($r,$gns);

        fclose($r);

    ?>

    <?php

        $rr = fopen("sample.txt","a");

        $gg = "hello this is second";

        fwrite($rr,$gg);

        //aa ma khali apped thayy che jayare ola ma prives remove thay ne bija parr lakhatuu nathuiiii...

    ?>

</body>

</html>

# **PHP File Upload**

## Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file\_uploads directive, and set it to On:

file\_uploads = On

## Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

<!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>

Some rules to follow for the HTML form above:

* Make sure that the form uses method="post"
* The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

* The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

The form above sends data to a file called "upload.php", which we will create next.

## Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

<?php  
$target\_dir = "uploads/";  
$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);  
$uploadOk = 1;  
$imageFileType = strtolower(pathinfo($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;  
  }  
}  
?>

PHP script explained:

* $target\_dir = "uploads/" - specifies the directory where the file is going to be placed
* $target\_file specifies the path of the file to be uploaded
* $uploadOk=1 is not used yet (will be used later)
* $imageFileType holds the file extension of the file (in lower case)
* Next, check if the image file is an actual image or a fake image

**Note:** You will need to create a new directory called "uploads" in the directory where "upload.php" file resides. The uploaded files will be saved there.

## Check if File Already Exists

Now we can add some restrictions.

First, we will check if the file already exists in the "uploads" folder. If it does, an error message is displayed, and $uploadOk is set to 0:

// Check if file already exists  
if (file\_exists($target\_file)) {  
  echo "Sorry, file already exists.";  
  $uploadOk = 0;  
}

## Limit File Size

The file input field in our HTML form above is named "fileToUpload".

Now, we want to check the size of the file. If the file is larger than 500KB, an error message is displayed, and $uploadOk is set to 0:

// Check file size  
if ($\_FILES["fileToUpload"]["size"] > 500000) {  
  echo "Sorry, your file is too large.";  
  $uploadOk = 0;  
}

## Limit File Type

The code below only allows users to upload JPG, JPEG, PNG, and GIF files. All other file types gives an error message before setting $uploadOk to 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;  
}

## Complete Upload File PHP Script

The complete "upload.php" file now looks like this:

<?php  
$target\_dir = "uploads/";  
$target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);  
$uploadOk = 1;  
$imageFileType = strtolower(pathinfo($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 ". htmlspecialchars( basename( $\_FILES["fileToUpload"]["name"])). " has been uploaded.";  
  } else {  
    echo "Sorry, there was an error uploading your file.";  
  }  
}  
?>

**Php sql**

    $servername = "localhost";

    $username = "root";

    $password = "";

    //create connection

    $con = mysqli\_connect($servername,$username,$password);

    if(!isset($con))

    {

        echo "not connect with database".mysqli\_connect\_error();

    }

    else{

        echo "connection is successfully";

}

Another method

//second method.botha are work.....

    $con = mysqli\_connect($servername,$username,$password);

    if($con->connect\_error)

    {

        die("connection fails".$con->connect\_error);

    }

    else{

        echo"hello connection is done";

    }

Create database

$sql = "CREATE DATABASE `databasecrud`"; //main

    if(mysqli\_query($con,$sql))

    {

        echo "create database succefully";

    }

    else

    {

        echo "database not created".mysqli\_error($con);

    }

Another method:-

    //its another method are workkk

    if ($con->query($sql) === TRUE) {

        echo "Database created successfully";

      } else {

        echo "Error creating database: " . $con->error;

      }

mysqli\_close($con); //close the conection insert after completing script….

**Advance Stuff:-**

**PHP mail() Function:-**

**What is PHP mail?**

PHP mail is the built in PHP function that is used to send emails from PHP scripts.

The mail function accepts the following parameters:

* Email address
* Subject
* Message
* CC or BC email addresses

## Why/When to use the mail PHP

### Sending mail using PHP

The PHP mail function has the following basic syntax

<?php

mail($to\_email\_address,$subject,$message,[$headers],[$parameters]);

?>

HERE,

* “$to\_email\_address” is the email address of the mail recipient
* “$subject” is the email subject
* “$message” is the message to be sent.
* “[$headers]” is optional, it can be used to include information such as CC, BCC
  + CC is the acronym for carbon copy. It’s used when you want to send a copy to an interested person i.e. a complaint email sent to a company can also be sent as CC to the complaints board.
  + BCC is the acronym for blind carbon copy. It is similar to CC. The email addresses included in the BCC section will not be shown to the other recipients.

## Simple Mail Transmission Protocol (SMTP)

PHP mailer uses Simple Mail Transmission Protocol (SMTP) to send mail.

On a hosted server, the SMTP settings would have already been set.

The SMTP mail settings can be configured from “php.ini” file in the PHP installation folder.

Configuring SMTP settings on your localhost Assuming you are using xampp on windows, locate the “php.ini” in the directory “C:\xampp\php”.

* Open it using notepad or any text editor. We will use notepad in this example. Click on the edit menu
* Click on Find… menu
* The find dialog menu will appear
* Click on Find Next button
* Locate the entries
  + *[mail function]*
  + *; XAMPP:* Don’t remove the semi column if you want to work with an SMTP Server like Mercury
  + ; SMTP = localhost
  + ; smtp\_port = 25
  + Remove the semi colons before SMTP and smtp\_port and set the SMTP to your [smtp server](https://www.guru99.com/free-smtp-servers.html) and the port to your smtp port. Your settings should look as follows
    - * SMTP = smtp.example.com
      * smtp\_port = 25
      * *Note the SMTP settings can be gotten from your web hosting providers.*
      * If the server requires authentication, then add the following lines.

auth\_username = example\_username@example.com

auth\_password = example\_password

Save the new changes.

Restart[Apache](https://www.guru99.com/apache.html)server.

**PHP Mail Example**

Let’s now look at an example that sends a simple mail.

<?php

$to\_email = 'name @ company . com';

$subject = 'Testing PHP Mail';

$message = 'This mail is sent using the PHP mail function';

$headers = 'From: noreply @ company . com';

mail($to\_email,$subject,$message,$headers);

?>

*Note: the above example only takes the 4 mandatory parameters.*

*You should replace the above fictitious email address with a real email address.*

**Sanitizing email user inputs**

The above example uses hard coded values in the source code for the email address and other details for simplicity.

Let’s assume you have to create a contact us form for users fill in the details and then submit.

* Users can accidently or intentional inject code in the headers which can result in sending spam mail
* To protect your system from such attacks, you can create a custom function that sanitizes and validates the values before the mail is sent.

Let’s create a custom function that validates and sanitizes the email address using the filter\_var built in function.

Filter\_var function The filter\_var function is used to sanitize and validate the user input data.

It has the following basic syntax.

<?php

filter\_var($field, SANITIZATION TYPE);

?>

HERE,

* “filter\_var(…)” is the validation and sanitization function
* “$field” is the value of the field to be filtered.
* “SANITIZATION TYPE” is the type of sanitization to be performed on the field such as;
  + **FILTER\_VALIDATE\_EMAIL** – it returns true for valid email addresses and false for invalid email addresses.
  + **FILTER\_SANITIZE\_EMAIL**– it removes illegal characters from email addresses. info\@domain.(com) returns info@domain.com.
  + **FILTER\_SANITIZE\_URL** – it removes illegal characters from URLs. http://www.example@.comé returns >http://www.example@.com
  + **FILTER\_SANITIZE\_STRING**– it removes tags from string values. <b>am bold</b> becomes am bold.

The code below implements uses a custom function to send secure mail.

<?php

function sanitize\_my\_email($field) {

$field = filter\_var($field, FILTER\_SANITIZE\_EMAIL);

if (filter\_var($field, FILTER\_VALIDATE\_EMAIL)) {

return true;

} else {

return false;

}

}

$to\_email = 'name @ company . com';

$subject = 'Testing PHP Mail';

$message = 'This mail is sent using the PHP mail ';

$headers = 'From: noreply @ company. com';

//check if the email address is invalid $secure\_check

$secure\_check = sanitize\_my\_email($to\_email);

if ($secure\_check == false) {

echo "Invalid input";

} else { //send email

mail($to\_email, $subject, $message, $headers);

echo "This email is sent using PHP Mail";

}

?>

## Secure Mail

Emails can be intercepted during transmission by unintended recipients.

This can exposure the contents of the email to unintended recipients.

Secure mail solves this problem by transmitting emails via Hypertext Transfer Protocol Secure (HTTPS).

HTTPS encrypts messages before sending them.

# PHP MySQLi Functions:

* [mysqli\_select\_db function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#2)
* [mysqli\_query function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#3)
* [mysqli\_num\_rows function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#4)
* [mysqli\_fetch\_array function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#5)
* [mysqli\_close function](https://www.guru99.com/mysql-php-and-other-database-access-methods.html#6)

# mysqli\_connect function

<?php;

$db\_server\_name = ‘localhost’

$db\_user\_name = ‘root’;

$db\_password =’’;

$db\_handle = mysqli\_connect($db\_server\_name, $db\_user\_name, $db\_password);

?>

* “$db\_handle” is the database connection resource variable.
* “mysqli\_connect(…)” is the function for php database connection
* “$server\_name” is the name or IP address of the server hosting MySQL server.
* “$user\_name” is a valid user name in MySQL server.
* “$password” is a valid password associated with a user name in MySQL server.

**mysqli\_select\_db($con,’mydb’);** //$con is a connection,,,and mydb is a database name

The mysqli\_select\_db function is used to select a database.

<?php

mysqli\_select\_db($con,$database\_name);

?>

* “mysqli\_select\_db(…)” is the database selection function that returns either true or false
* “$database\_name” is the name of the database
* “$link\_identifier” is optional, it is used to pass in the server connection link

## PHP mysqli\_query function

The mysqli\_query function is used to execute[SQL](https://www.guru99.com/sql.html)queries.

The function can be used to execute the following query types;

* Insert
* Select
* Update
* delete

It has the following syntax.

<?php

mysqli\_query($db\_handle,$query) ;

?>

## PHP mysqli\_num\_rows function

The mysqli\_num\_rows function is used to get the number of rows returned from a select query.

It has the following syntax.

<?php

mysqli\_num\_rows($result);

?>

HERE,

* “mysqli\_num\_rows(…)” is the row count function
* “$result” is the mysqli\_query result set

## PHP mysqli\_close function

The mysqli\_close function is used to close an open database connection.

It has the following syntax.

<?php

mysqli\_close($conx);

?>

**OOPS:-**

OOP stands for Object-Oriented Programming.

* OOP is faster and easier to execute
* OOP provides a clear structure for the programs
* OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
* OOP makes it possible to create full reusable applications with less code and shorter development time

## Define a Class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}). All its properties and methods go inside the braces:

### **Syntax**

<?php  
class Fruit {  
  // code goes here...  
}  
?>

**OBJECT :-**

$apple = new Fruit();

### **Example**

<?php  
class Fruit {  
  // Properties  
  public $name;  
  public $color;  
  
  // Methods  
  function set\_name($name) {  
    $this->name = $name;  
  }  
  function get\_name() {  
    return $this->name;  
  }  
  function set\_color($color) {  
    $this->color = $color;  
  }  
  function get\_color() {  
    return $this->color;  
  }  
}  
  
$apple = new Fruit();  
$apple->set\_name('Apple');  
$apple->set\_color('Red');  
echo "Name: " . $apple->get\_name();  
echo "<br>";  
echo "Color: " . $apple->get\_color();  
?>

## PHP - The $this Keyword

The $this keyword refers to the current object, and is only available inside methods.

## PHP - instanceof

You can use the instanceof keyword to check if an object belongs to a specific class:

### **Example**

<?php  
$apple = new Fruit();  
var\_dump($apple instanceof Fruit);  
?>

Output:-

bool(true)

## PHP - The \_\_construct Function

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (\_\_)!

We see in the example below, that using a constructor saves us from calling the set\_name() method which reduces the amount of code:

### **Example**

<?php  
class Fruit {  
  public $name;  
  public $color;  
  
  function \_\_construct($name) {  
    $this->name = $name;  
  }  
  function get\_name() {  
    return $this->name;  
  }  
}  
  
$apple = new Fruit("Apple");  
echo $apple->get\_name();  
?>

Myexample:-

<?php

class frut{

    public $name;

    public $type;

    function set\_value($name,$type){

        $this->name = $name;

        $this->type = $type;

    }

    function get\_value(){

        return $this->name .' '.$this->type.'<BR>';

    }

    function \_\_construct(){

        echo "hello this is construct...";  //its autometicaly wrok start  of the script

    }

    function \_\_destruct(){

        echo"hello this is exits functiono...."; //its autometicaly wrok end of the script

    }

}

$aa = new frut();

$aa1 = new frut();

$aa->set\_value('apple','red');

$aa1->set\_value('banana','yellow');

echo $aa->get\_value();

echo $aa1->get\_value();

var\_dump($aa instanceof frut); // its use to only check this object is releted to those class..

?>

## PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

* public - the property or method can be accessed from everywhere. This is default
* protected - the property or method can be accessed within the class and by classes derived from that class
* private - the property or method can ONLY be accessed within the class

### **Example**

<?php  
class Fruit {  
  public $name;  
  protected $color;  
  private $weight;  
}  
  
$mango = new Fruit();  
$mango->name = 'Mango'; // OK  
$mango->color = 'Yellow'; // ERROR  
$mango->weight = '300'; // ERROR  
?>

In the next example we have added access modifiers to two functions. Here, if you try to call the set\_color() or the set\_weight() function it will result in a fatal error (because the two functions are considered protected and private), even if all the properties are public:

### **Example**

<?php  
class Fruit {  
  public $name;  
  public $color;  
  public $weight;  
  
  function set\_name($n) {  // a public function (default)  
    $this->name = $n;  
  }  
  protected function set\_color($n) { // a protected function  
    $this->color = $n;  
  }  
  private function set\_weight($n) { // a private function  
    $this->weight = $n;  
  }  
}  
  
$mango = new Fruit();  
$mango->set\_name('Mango'); // OK  
$mango->set\_color('Yellow'); // ERROR  
$mango->set\_weight('300'); // ERROR  
?>

Myexample:-

<?php

class frut{

        protected $name = 'ganesh'; //prtotectecd method use class and on herited class both

        private $type = "hello";

        function set\_value($name,$type){

            $this->name = $name;

            $this->type = $type;

        }

        function get\_value(){

            return $this->name .' '.$this->type.'<BR>';

        }

        // function \_\_construct(){

        //     echo "hello this is construct...";  //its autometicaly wrok start  of the script

        // }

        // function \_\_destruct(){

        //     echo"hello this is exits functiono...."; //its autometicaly wrok end of the script

        // }

}

class vegitable extends frut{

        public $vname ='raval';

        public $vtype = 'how are';

        function set\_value\_v($vname,$vtype){

            $this->vname = $vname;

            $this->vtype = $vtype;

        }

        function get\_value\_v(){

            return $this->name;

        }

        function get\_val()

        {

            return $this->name;

        }

        // function \_\_construct(){

        //     echo "hello this is construct...";  //its autometicaly wrok start  of the script

        // }

        // function \_\_destruct(){

        //     echo"hello this is exits functiono...."; //its autometicaly wrok end of the script

        // }

}

$aa = new vegitable();

$aa1 = new vegitable();

// $aa->set\_value('apple','helo');

// $aa1->set\_value\_v('banana','yellow');// its overriding method

echo $aa->get\_value();

echo $aa1->get\_value\_v();

echo $aa1->get\_val();

?>

## PHP - Class Constants

Constants cannot be changed once it is declared.

Class constants can be useful if you need to define some constant data within a class.

A class constant is declared inside a class with the const keyword.

Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

We can access a constant from outside the class by using the class name followed by the scope resolution operator (::) followed by the constant name, like here:

### **Example**

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
}  
  
echo Goodbye::LEAVING\_MESSAGE;  
?>

Or, we can access a constant from inside the class by using the self keyword followed by the scope resolution operator (::) followed by the constant name, like here:

### **Example**

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
  public function byebye() {  
    echo self::LEAVING\_MESSAGE;  
  }  
}  
  
$goodbye = new Goodbye();  
$goodbye->byebye();  
?>

**My example:-**

<?php

    class hello{

        const am = "hello how are you??";

        function abc(){

            echo "this is function";

            echo self::am;

        }

    }

    echo hello::am;

    $aa = new hello();

    $aa->abc();

?>

## PHP - What are Traits?

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

Traits are used to declare methods that can be used in multiple classes. Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier (public, private, or protected).

Traits are declared with the trait keyword:

### **Syntax**

<?php  
trait TraitName {  
  // some code...  
}  
?>

To use a trait in a class, use the use keyword:

### **Syntax**

<?php  
class MyClass {  
  use TraitName;  
}  
?>

Let's look at an example:

### **Example**

<?php  
trait message1 {  
public function msg1() {  
    echo "OOP is fun! ";  
  }  
}  
  
class Welcome {  
  use message1;  
}  
  
$obj = new Welcome();  
$obj->msg1();  
?>

## PHP - Using Multiple Traits

Let's look at another example:

### **Example**

<?php

    trait abc{

        function ave(){

            echo "hello abc";

        }

    }

    trait xyz{

       function ave1(){

        echo "hello xyz";

       }

    }

    trait pqr{

       function ave2(){

        echo "hello pqr";

       }

    }

    class hello{

        const am = "hello how are you??";

        function abc(){

            echo "this is function";

            echo self::am;

        }

        use xyz;

        use pqr;

        use abc;

    }

    echo hello::am;

    $aa = new hello();

    $aa->ave(); echo "<br>";

    $aa->ave1();

    $aa->ave2();

    //$aa->xyz();

?>

## PHP - Static Methods

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the static keyword:

### **Syntax**

<?php  
class ClassName {  
  public static function staticMethod() {  
    echo "Hello World!";  
  }  
}  
?>

To access a static method use the class name, double colon (::), and the method name:

### **Syntax**

ClassName::staticMethod();

Let's look at an example:

### **Example**

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
}  
  
// Call static method  
greeting::welcome();  
?>

**//aama and static variavle ma ajj diffrenece che only ke aema only varieable $thi thay and call pan $thiijj thayy and aama aveuu na hoy**

**//aa and const are similar but aema jayare apde another function ma use kariye taye apde self:: kari ne use kariye chiye**

**//jayar amaa only :: amamhh jj use thsase**

## PHP - Static Properties

Static properties can be called directly - without creating an instance of a class.

Static properties are declared with the static keyword:

### **Syntax**

<?php  
class ClassName {  
  public static $staticProp = "W3Schools";  
}  
?>

### **Syntax**

ClassName::$staticProp;

Let's look at an example:

### **Example**

<?php  
class pi {  
  public static $value = 3.14159;  
}  
  
// Get static property  
echo pi::$value;  
?>

### **Example**

<?php  
class pi {  
  public static $value=3.14159;  
}  
  
class x extends pi {  
  public function xStatic() {  
    return parent::$value;  
  }  
}  
  
// Get value of static property directly via child class  
echo x::$value;  
  
// or get value of static property via xStatic() method  
$x = new x();  
echo $x->xStatic();  
?>

# **PHP Namespaces:-**

Namespaces are qualifiers that solve two different problems:

1. They allow for better organization by grouping classes that work together to perform a task
2. They allow the same name to be used for more than one class

For example, you may have a set of classes which describe an HTML table, such as Table, Row and Cell while also having another set of classes to describe furniture, such as Table, Chair and Bed. Namespaces can be used to organize the classes into two different groups while also preventing the two classes Table and Table from being mixed up.

## Declaring a Namespace

Namespaces are declared at the beginning of a file using the namespace keyword:

### **Syntax**

Declare a namespace called Html:

<?php  
namespace Html;  
?>

**Note:** A namespace declaration must be the first thing in the PHP file. The following code would be invalid:

Myeample:-

Test.php:-

<?php

    namespace test;

    function abc(){

        echo "hello one";

    }

?>

Test2.php:-

<?php

    namespace test2;

    function abc(){

        echo "hello two";

    }

?>

Index.php:-

<?php

    include 'test.php';

    include 'test2.php';

    test\abc(); //its calling function in test namespace

    test2\abc(); //jayare same name na function ke class hoy tayare apde namespace no use kariye chiye..

?>

And another class use in namespance…

Test.php:-

<?php

    namespace test;

    class ganesh{

        function abc(){

            echo "hello one";

        }

    }

?>

Test2.php

<?php

    namespace test2;

    class ganesh{

        function abc(){

            echo "hello two";

        }

        function hello(){

            echo "hello how are you";

        }

    }

?>

Index.php:-

<?php

    include 'test.php';

    include 'test2.php';

    // test\abc(); //its calling function in test namespace

    // test2\abc(); //jayare same name na function ke class hoy tayare apde namespace no use kariye chiye..

    // echo "<br>";

    $obj = new test\ganesh;

    $obj->abc();

    $obj2 = new test2\ganesh;

    $obj2->hello();

?>

## PHP - What is an Iterable?

An iterable is any value which can be looped through with a foreach() loop.

The iterable pseudo-type was introduced in PHP 7.1, and it can be used as a data type for function arguments and function return values.

## PHP - Using Iterables

The iterable keyword can be used as a data type of a function argument or as the return type of a function:

### **Example**

Use an iterable function argument:

<?php  
function printIterable(iterable $myIterable) {  
  foreach($myIterable as $item) {  
    echo $item;  
  }  
}  
  
$arr = ["a", "b", "c"];  
printIterable($arr);  
?>

### **Example**

Return an iterable:

<?php  
function getIterable():iterable {  
  return ["a", "b", "c"];  
}  
  
$myIterable = getIterable();  
foreach($myIterable as $item) {  
  echo $item;  
}  
?>

PHP - Creating Iterables

**Arrays**

All arrays are iterables, so any array can be used as an argument of a function that requires an iterable.

**Iterators**

Any object that implements the Iterator interface can be used as an argument of a function that requires an iterable.

An iterator contains a list of items and provides methods to loop through them. It keeps a pointer to one of the elements in the list. Each item in the list should have a key which can be used to find the item.

An iterator must have these methods:

* current() - Returns the element that the pointer is currently pointing to. It can be any data type
* key() Returns the key associated with the current element in the list. It can only be an integer, float, boolean or string
* next() Moves the pointer to the next element in the list
* rewind() Moves the pointer to the first element in the list
* valid() If the internal pointer is not pointing to any element (for example, if next() was called at the end of the list), this should return false. It returns true in any other case

<?php  
// Create an Iterator  
class MyIterator implements Iterator {  
  private $items = [];  
  private $pointer = 0;  
  
  public function \_\_construct($items) {  
    // array\_values() makes sure that the keys are numbers  
    $this->items = array\_values($items);  
  }  
  
  public function current() {  
    return $this->items[$this->pointer];  
  }  
  
  public function key() {  
    return $this->pointer;  
  }  
  
  public function next() {  
    $this->pointer++;  
  }  
  
  public function rewind() {  
    $this->pointer = 0;  
  }  
  
  public function valid() {  
    // count() indicates how many items are in the list  
    return $this->pointer < count($this->items);  
  }  
}  
  
// A function that uses iterables  
function printIterable(iterable $myIterable) {  
  foreach($myIterable as $item) {  
    echo $item;  
  }  
}  
  
// Use the iterator as an iterable  
$iterator = new MyIterator(["a", "b", "c"]);  
printIterable($iterator);  
?>

## What is XML?

The XML language is a way to structure data for sharing across websites.

Several web technologies like RSS Feeds and Podcasts are written in XML.

XML is easy to create. It looks a lot like HTML, except that you make up your own tags.

If you want to learn more about XML, please visit our [XML tutorial](https://www.w3schools.com/xml/default.asp).

What is an XML Parser?

To read and update, create and manipulate an XML document, you will need an XML parser.

In PHP there are two major types of XML parsers:

* Tree-Based Parsers
* Event-Based Parsers

Tree-Based Parsers

Tree-based parsers holds the entire document in Memory and transforms the XML document into a Tree structure. It analyzes the whole document, and provides access to the Tree elements (DOM).

This type of parser is a better option for smaller XML documents, but not for large XML document as it causes major performance issues.

Example of tree-based parsers:

* SimpleXML
* DOM

Event-Based Parsers

Event-based parsers do not hold the entire document in Memory, instead, they read in one node at a time and allow you to interact with in real time. Once you move onto the next node, the old one is thrown away.

This type of parser is well suited for large XML documents. It parses faster and consumes less memory.

Example of event-based parsers:

* XMLReader
* XML Expat Parser

## What is JSON?

JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.

Since the JSON format is a text-based format, it can easily be sent to and from a server, and used as a data format by any programming language.

PHP and JSON

PHP has some built-in functions to handle JSON.

First, we will look at the following two functions:

* json\_encode()
* json\_decode()

## PHP - json\_encode()

The json\_encode() function is used to encode a value to JSON format.

### **Example**

This example shows how to encode an associative array into a JSON object:

<?php  
$age = array("Peter"=>35, "Ben"=>37, "Joe"=>43);  
  
echo json\_encode($age);  
?>

### **Example**

This example shows how to encode an indexed array into a JSON array:

<?php  
$cars = array("Volvo", "BMW", "Toyota");  
  
echo json\_encode($cars);  
?>

## PHP - json\_decode()

The json\_decode() function is used to decode a JSON object into a PHP object or an associative array.

### **Example**

This example decodes JSON data into a PHP object:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
var\_dump(json\_decode($jsonobj));  
?>

The json\_decode() function returns an object by default. The json\_decode() function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.

### **Example**

This example decodes JSON data into a PHP associative array:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
var\_dump(json\_decode($jsonobj, true));  
?>

## PHP - Accessing the Decoded Values

Here are two examples of how to access the decoded values from an object and from an associative array:

### **Example**

This example shows how to access the values from a PHP object:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
$obj = json\_decode($jsonobj);  
  
echo $obj->Peter;  
echo $obj->Ben;  
echo $obj->Joe;  
?>

### **Example**

This example shows how to access the values from a PHP associative array:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
$arr = json\_decode($jsonobj, true);  
  
echo $arr["Peter"];  
echo $arr["Ben"];  
echo $arr["Joe"];  
?>

## PHP - Looping Through the Values

You can also loop through the values with a foreach() loop:

### **Example**

This example shows how to loop through the values of a PHP object:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
$obj = json\_decode($jsonobj);  
  
foreach($obj as $key => $value) {  
  echo $key . " => " . $value . "<br>";  
}  
?>

### **Example**

This example shows how to loop through the values of a PHP associative array:

<?php  
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';  
  
$arr = json\_decode($jsonobj, true);  
  
foreach($arr as $key => $value) {  
  echo $key . " => " . $value . "<br>";  
}  
?>

Simple my form xml:-

Create one xml file:-

<?xml version="1.0" encoding ="utf-8"?>

<employees status="ok">

    <record man\_no = "101">

        <name>joe paul</name>

        <position>ceo</position>

    </record>

    <record man\_no = "102">

        <name>Tasha Smith</name>

        <position>Finance Manager</position>

    </record>

</employees>

And second create file php

And create php file and to read xml file

<?php

    $xml = simplexml\_load\_file('main.xml');

    echo '<h2>employee listing</h2>';

    $list = $xml->record;

    for ($i = 0;$i<count($list);$i++)

    {

        echo 'man number:-'.$list[$i]->attributes()->man\_no."<br>";

        echo 'name:-'.$list[$i]->name."<br>";

        echo 'position:-'.$list[$i]->position."<br>";

        echo "<br><br>";

    }

?>

Another method:-

Using dom to create xml file :-

<?php

    $dom = new DOMDocument();

        $dom->encoding = 'utf-8';

        $dom->xmlVersion = '1.0';

        $dom->formatOutput = true;

        $xmlfile1 = 'main3.xml';

        $roots = $dom->createElement('univercity');

        $clg\_node = $dom->createElement('college');

        $clg\_att = new DOMAttr('id','101');

        $clg\_node->setAttributeNode($clg\_att);

        $clg\_name = $dom->createElement('name','gls');

        $clg\_node->appendChild($clg\_name);

        $clg\_noof = $dom->createElement('noof','100');

        $clg\_node->appendChild($clg\_noof);

        $clg\_rank = $dom->createElement('rank','01');

        $clg\_node->appendChild($clg\_rank);

        $clg\_medum = $dom->createElement('medum','english');

        $clg\_node->appendChild($clg\_medum);

        $roots->appendChild($clg\_node);

        $dom->appendChild($roots);

        $dom->save($xmlfile1);

        echo "is succefull run";

?>

## PHP JSON Introduction

The JSON extension implements the JavaScript Object Notation data-interchange format.

## PHP JSON Functions

|  |  |
| --- | --- |
| **Function** | **Description** |
| [json\_decode()](https://www.w3schools.com/php/func_json_decode.asp) | Decodes a JSON string |
| [json\_encode()](https://www.w3schools.com/php/func_json_encode.asp) | Encode a value to JSON format |
| [json\_last\_error()](https://www.w3schools.com/php/func_json_last_error.asp) | Returns the last error occurred |
| json\_last\_error\_msg() | Returns the error string of the last  json\_encode() or json\_decode() call |

<?php

    //encode..

    $age = array("pater"=>35,"ben"=>34,"joe"=>34);

    echo json\_encode($age)."<br>";

    $age1 = '{"pater":35,"ben":34,"joe":34}';

    var\_dump(json\_decode($age1,true));

    echo "<br>";

    $result = json\_decode($age1);

    echo $result->pater;

    $result1 = json\_decode($age1,true);

    echo "<br>";

    echo $result1['ben'];

?>