PHP Web Development, Lesson 1

Contents

[1 PHP Tutorial 1](#_Toc194742949)

[2 PhpStorm - IDE 2](#_Toc194742950)

[2.1 Download 2](#_Toc194742951)

[2.2 Usage 2](#_Toc194742952)

[3 Processing Form Data, Examples 4](#_Toc194742953)

[3.1 Example 1 - POST 4](#_Toc194742954)

[3.2 Example 2 - Numbers 5](#_Toc194742955)

[3.3 Example 3 - Formatting 6](#_Toc194742956)

[3.4 Example 4 - Checkboxes 7](#_Toc194742957)

[3.5 Example 5 - Functions 8](#_Toc194742958)

[3.6 Example 6 - Includes 9](#_Toc194742959)

[3.7 Example 7 - GET Example 10](#_Toc194742960)

[3.8 When to use GET and POST 11](#_Toc194742961)

[Appendix 1 W3Schools – PHP 12](#_Toc194742962)

[Appendix 1.1 Introduction 12](#_Toc194742963)

[Appendix 1.2 Syntax 12](#_Toc194742964)

[Appendix 1.3 Comments 13](#_Toc194742965)

[Appendix 1.4 Variables 13](#_Toc194742966)

[Appendix 1.5 Variable Scope 14](#_Toc194742967)

[Appendix 1.6 Output 15](#_Toc194742968)

[Appendix 1.7 Strings 15](#_Toc194742969)

[Appendix 1.8 Numbers 16](#_Toc194742970)

[Appendix 1.9 Misc 17](#_Toc194742971)

# PHP Tutorial

[PHP Tutorial](https://www.w3schools.in/php/tutorials/) – Simple break down of 64 topics/use cases.

# PhpStorm - IDE

## Download

Get a free educational license from JetBrains (company that makes PhpStorm):

<https://www.jetbrains.com/community/education/#students>

Probably, in the process above, you will find a place to download PhpStorm (or any of their other products).

## Usage

This is a Hello World PHP app to show how to create a page in PhpStorm and display it on *localhost*.

1. Create a folder in your www folder (probably located at: C:\wamp64\www). Perhaps name it: *hw3*.
2. Start PhpStorm, choose the hamburger menu (upper-left), and then choose: File, New Project, PHP Empty Project
3. Navigate to the folder you created above in the “Location” field and then choose: Create.
4. Select the “hw3” node, right-click and choose: New, PHP File, and give it a name, *helloworld.php*.
5. Paste this code in (delete over what is there) and save.

<?php

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

echo "<h1>Hello World!</h1>";

$date = date("D, m.d.Y, h:i:s");

echo "<p>$date</p>";

?>



1. Run WAMP (or MAMP)
2. Launch localhost (See Lab 1 if necessary)
3. In the URL, type: *localhost/hw3/* and press Enter. You’ll see the files in your *hw3* folder. Select *helloworld.php* and it will display.

|  |  |
| --- | --- |
|  |  |

1. You can make changes to the page in PhpStorm, and then simply refresh the page in the browser and the changes will show. This usually works. Sometimes it seems to use a cached version – so, if things don’t appear to be working, type the URL in again and enter. Add this code to the bottom of your file (above the “?>” closing PHP tag) and then go to *localhost* and refresh your page.

$s1 = "horse";

$s2 = "pig";

$s3 = "rhino";

$strs = array($s1, $s2, $s3);

$i = 1;

foreach ($strs as $str) {

 echo strval($i) . ". " . $str . "<br>";

 $i++;

}



Notes:

* There is a Run menu, but it doesn’t seem to work.
* The Debugger does work…sometimes. Set breakpoint, Choose: Hamburger, Run, Debug, choose the file, launch page in localhost.
* You may need to set the PHP interpreter: Hamburger, Run, Edit Configurations, PHP Script (on left), Select file, and then set Interpreter by navigating to php.exe: C:\wamp64\bin\php\php8.4.0.

# Processing Form Data, Examples

## Example 1 - POST

1. Topics: (a) php tags, (b) variable names, (c) global variables, (d) form: action & method, (e) submit button.

PHP uses the HTML *name* attribute to access variables that have been posted (as opposed to the *id* attribute). The [reason](https://stackoverflow.com/questions/69135131/why-does-phps-post-variable-get-values-from-htmls-name-attribute-instead-of).

*ex1-post.php*

<!DOCTYPE html>
<html lang="en">
<head>
 <title>PHP POST Example</title>
 <?php
 date\_default\_timezone\_set('America/New\_York');
 $curr\_date = date("l, m.d.Y, h:i:s");
 ?>
 <link rel="stylesheet" href="site.css">
</head>
<body>

<h2>PHP POST Example</h2>
<p>
 <?php
 echo $curr\_date;
 ?>
</p>
<form action="index\_post\_results.php" method="POST">
 <p>Name: <input type="text" name="name"></p>
 <p>E-mail: <input type="text" name="email"></p>
 <input type="submit">
</form>
</body>
</html>

*ex1-results.php*

<!DOCTYPE html>
<html lang="en">
<head>
 <title>PHP POST Results</title>
 <?php
 date\_default\_timezone\_set('America/New\_York');
 $curr\_date = date("l, m.d.Y, h:i:s");
 ?>
 <link rel="stylesheet" href="site.css">
</head>
<body>
<h2>PHP POST Results</h2>
<p>
 The values you entered on: <?php echo $curr\_date; ?>
</p>
<p>Welcome <span class="keyword"><?php echo $\_POST["name"]; ?></span>,
Your email address is: <span class="keyword">

 <?php echo $\_POST["email"]; ?></span>.</p>

</body>
</html>

1. Displaying the date in *ex1-post* above can also be done as shown below. Generally, prefer not to code the HTML, but sometimes it is needed. Topics: (a) variable substitution, (c) string concatenation.

<?php
 echo "<p>$curr\_date</p>";
 echo "<p>" . $curr\_date . "</p>";
?>

## Example 2 - Numbers

1. Topic: type inference

*ex2-numbers.php*

<h2>Ex 2 - Numbers</h2>
<form action="ex2-results.php" method="POST">
 <p>Value 1: <input type="text" name="val1"></p>
 <p>Value 2: <input type="text" name="val2"></p>
 <input type="submit">
</form>

*ex2-results.php*

<body>
<?php
 $val1 = $\_POST["val1"];
 $val2 = $\_POST["val2"];
 $product = $val1 \* $val2;
?>
<h2>Ex 2 - Numbers Results</h2>
<p>
 The product is: <?php echo $val1; ?> \* <?php echo $val2; ?> =

 <?php echo $product; ?>
</p>
</body>

1. Topics: var\_dump & type inference

$val1 = $\_POST["val1"];
var\_dump($val1);
$val2 = $\_POST["val2"];
var\_dump($val2);
$product = $val1 \* $val2;
var\_dump($product);

1. Can cast to be double (float):

$val1 = doubleval($\_POST["val1"]);
var\_dump($val1);
$val2 = doubleval($\_POST["val2"]);
var\_dump($val2);

There is an *is\_double* function also.

## Example 3 - Formatting

1. Topics: (a) formatting strings, (b) arrays, (c) foreach-looping over array, (d) display results in text area and table

*ex3-results.php* (see also: *ex3-formatting*)

<body>
<?php

$val1 = doubleval($\_POST["val1"]);
$val2 = doubleval($\_POST["val2"]);
$vals = array($val1, $val2);
$msg = "";
$i = 0;
foreach ($vals as $val) {
 $msg .= sprintf("Value %d=%.2f, raw=%s\n", ++$i, $val, $val);
}

?>
<h2>Ex 3 - Formatting Results</h2>

<textarea rows="10" cols="50"><?php echo $msg;?>

</textarea>

<table class="stats\_table">
 <tr>
 <th>Value</th>
 <th>Raw</th>
 </tr>
 <tr>
 <td><?php echo sprintf("%.2f", $val1); ?></td>
 <td><?php echo $val1 ?></td>
 </tr>
 <tr>
 <td><?php echo sprintf("%.2f", $val2); ?></td>
 <td><?php echo $val2 ?></td>
 </tr>
</table>

</body>

## Example 4 - Checkboxes

1. Topics: (a) defining checkbox array, (b) check if value was posted, (c) count, (d) indexed loop, (e) substring

***ex4-checkboxes.php***

<form action="ex4\_results.php" method="POST">
 <p>Pick your favorite sports:</p>
 <p>
 <input type="checkbox" name="sports[]" id="c1" value="Baseball"/>
 <label for="">Baseball</label>
 <input type="checkbox" name="sports[]" id="c2" value="Basketball"/>
 <label for="">Basketball</label>
 <input type="checkbox" name="sports[]" id="c3" value="Football"/>
 <label for="">Football</label>
 <input type="checkbox" name="sports[]" id="c4" value="Soccer"/>
 <label for="">Soccer</label>
 </p>
 <input type="submit" >
</form>



***ex4-results.php***

<body>
<h2>Ex 4 - Checkbox Results</h2>
<?php
if(!array\_key\_exists("sports", $\_POST)) {
 echo("--> You didn't select any sports.");
}
else {
 $sports = $\_POST['sports'];
 $n = count($sports);
 $msg = "<p>";
 if($n == 1) {
 $msg .= "You like this sport: ";
 }
 else {
 $msg .= "You like these sports: ";
 }
 for($i=0; $i<$n; $i++) {
 *//$msg .= $sports[$i] . ", ";* $msg = "$msg $sports[$i], ";
 }
 $msg = substr($msg, 0, -2);
 $msg .= "</p>";
 echo $msg;
}
?>
</body>

## Example 5 - Functions

1. Topics: (a) defining radio button array, (b) functions, (e) global variables

***ex5-functions.php***

<form action="ex5-results.php" method="POST">
 <p>Pick your favorite sport:</p>
 <p>
 <input type="radio" name="sports[]" id="c1" value="Baseball"/>
 <label for="">Baseball</label>
 <input type="radio" name="sports[]" id="c2" value="Basketball"/>
 <label for="">Basketball</label>
 <input type="radio" name="sports[]" id="c3" value="Football"/>
 <label for="">Football</label>
 <input type="radio" name="sports[]" id="c4" value="Soccer"/>
 <label for="">Soccer</label>
 </p>
 <input type="submit" >
</form>

***ex5-results.php***

<body>
<h2>Ex 4 - Checkbox Results</h2>
<?php
function sports\_message($sports) {
 $msg = "You like this sport: $sports[0]";
 return $msg;
}
function make\_paragraph($msg) {
 $msg = "<p>$msg</p>";
 return $msg;
}
function sports\_message2() {
 global $sports;
 $msg = "You like this sport: $sports[0]";
 return $msg;
}
?>
<?php
if(!array\_key\_exists("sports", $\_POST)) {
 echo("--> You didn't select any sports.");
}
else {
 $sports = $\_POST['sports'];
 $msg = sports\_message($sports);
 $msg = make\_paragraph($msg);
 echo $msg;
}
?>
</body>
</html>

## Example 6 - Includes

1. Topics: (a) defining radio button array, (b) functions, (e) global variables

***ex6-includes.php***

<form action="ex6-results.php" method="POST">
 <p>Pick your favorite sports:</p>
 <p>
 <select name="sports[]" size="5" multiple>
 <option value="Baseball">Baseball</option>
 <option value="Basketball">Basketball</option>
 <option value="Football">Football</option>
 <option value="Soccer">Soccer</option>
 </select>
 </p>
 <input type="submit" >
</form>

***ex6-code.php***

<?php
function display\_sports() {
 $sports = $\_POST['sports'];
 $n = count($sports);
 $msg = "<p>";
 if($n == 1) {
 $msg .= "You like this sport: ";
 }
 else {
 $msg .= "You like these sports: ";
 }
 for($i=0; $i<$n; $i++) {
 *//$msg .= $sports[$i] . ", ";* $msg = "$msg $sports[$i], ";
 }
 $msg = substr($msg, 0, -2);
 $msg .= "</p>";
 echo $msg;
}
?>

***ex6-results.php***

<head>
 <title>Ex 6-Includes Results</title>
 <link rel="stylesheet" href="site.css">
 <?php
 include "ex6-code.php";
 ?>
</head>
<body>
<h2>Ex 6 - Includes Results</h2>
<?php
if(!array\_key\_exists("sports", $\_POST)) {
 echo("--> You didn't select any sports.");
}
else {
 display\_sports();
}
?>
</body>

## Example 7 - GET Example

Topic: GET

***ex7\_get.php***

<!DOCTYPE html>
<html lang="en">
<head>
 <title>PHP GET Example</title>
 <link rel="stylesheet" href="site.css">
</head>
<body>

<h2>PHP GET Example</h2>
<form action="index\_get\_results.php" method="GET">
 <p>Name: <input type="text" name="name"></p>
 <p>E-mail: <input type="text" name="email"></p>
 <input type="submit">
</form>

</body>
</html>

***ex7\_results.php***

<!DOCTYPE html>
<html lang="en">
<head>
 <title>PHP GET Results</title>
 <?php
 date\_default\_timezone\_set('America/New\_York');
 $curr\_date = date("l, m.d.Y, h:i:s");
 ?>
 <link rel="stylesheet" href="site.css">
</head>
<body>
<h2>PHP GET Results</h2>
<p>
 The values you entered on: <?php echo $curr\_date; ?>
</p>
<p>Welcome <span class="keyword"><?php echo $\_GET["name"]; ?></span>,
Your email address is: <span class="keyword"><?php echo $\_GET["email"]; ?></span>.</p>

</body>
</html>

## When to use GET and POST

Source: can’t find it. This is a direct copy:

According to the HTTP specification, you should use the POST method when you're using the form to change the state of something on the server end. For example, if a page has a form to allow users to add their own comments, the form should use POST. If you click "Reload" or "Refresh" on a page that you reached through a POST, it's almost always an error -- you shouldn't be posting the same comment twice -- which is why these pages aren't bookmarked or cached.

You should use the GET method when your form is, well, getting something off the server and not actually changing anything. For example, the form for a search engine should use GET, since searching a Web site should not be changing anything that the client might care about, and bookmarking or caching the results of a search-engine query is just as useful as bookmarking or caching a static HTML page.

Appendix

1. W3Schools – PHP
	1. Introduction

Source: <https://www.w3schools.com/php/php_intro.asp>

What is PHP?

* PHP is an acronym for "PHP: Hypertext Preprocessor"
* PHP is a widely-used, open source scripting language: Facebook, Wikipedia, Tumblr, Slack, Etsy, Lyft, WordPress, *etc.*
* PHP scripts are executed on the server
* PHP is free to download and use

What is a PHP File?

* PHP files can contain text, HTML, CSS, JavaScript, and PHP code
* PHP code is executed on the server, and the result is returned to the browser as plain HTML
* PHP files have extension ".php"
	1. Syntax

Source: <https://www.w3schools.com/php/php_syntax.asp>

* A PHP script can be placed anywhere in the document.
* A PHP script starts with <?php and ends with ?>
* The default file extension for PHP files is ".php".
* A PHP file normally contains HTML tags, and some PHP scripting code.
* In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive. However; all variable names are case-sensitive!

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

**<?php**

echo "Hello World!";

**?>**

</body>

</html>

* 1. Comments
* Single line comments start with //.
* You can also use # for single line comments, but in this tutorial we will use //.
* Multi-line comments start with /\* and end with \*/.
* The multi-line comment syntax can also be used to prevent execution of parts inside a code-line

$x = 5 /\* + 15 \*/ + 5;

echo $x;

* 1. Variables
* In PHP, a variable starts with the $ sign, followed by the name of the variable PHP has no command for declaring a variable. It is created the moment you first assign a value to it.
* PHP is a Loosely Typed Language
* PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
* In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

$x = 5; // $x is an integer

$y = "John"; // $y is a string

echo $x;

echo $y;

* PHP supports the following data types: String, Integer, Float, Boolean, Array, Object, NULL, Resource
* To get the data type of a variable, use the var\_dump() function.

|  |  |
| --- | --- |
| **Code** | **Output** |
| var\_dump(5);var\_dump("John");var\_dump(3.14);var\_dump(true);var\_dump([2, 3, 56]);var\_dump(NULL); | int(5)string(4) "John"float(3.14)bool(true)array(3) { [0]=> int(2) [1]=> int(3) [2]=> int(56)}NULL |

* You can assign the same value to multiple variables in one line:

$x = $y = $z = "Fruit"

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

$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>";

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

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>";

* The global keyword is used to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function):

$x = 5;

$y = 10;

function myTest() {

 global $x, $y;

 $y = $x + $y;

}

myTest();

echo $y; // outputs 15

* PHP also stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly. The example above can be rewritten like this:

$x = 5;

$y = 10;

function myTest() {

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

}

myTest();

echo $y; // outputs 15

* Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the *static* keyword when you first declare the variable:

function myTest() {

 static $x = 0;

 echo $x;

 $x++;

}

myTest(); // outputs 0

myTest(); // outputs 1

myTest(); // outputs 2

* 1. Output
* With PHP, there are two basic ways to get output: echo and print. 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. Either statement can be used with or without parentheses:

echo "<h2>PHP is Fun!</h2>";

print "Hello world!<br>";

echo ("I'm about to learn PHP!<br>");

print ("This ", "string ", "has ", "with multiple parameters.");

* 1. Strings
* You can use double or single quotes, but you should be aware of the differences between the two.

Double quoted string literals perform operations for special characters:

$x = "John";

echo "Hello $x";

* Concatenate:

$x = "Hello";

$y = "World";

$z = $x . " " . $y;

echo $z;

* Length
* echo strlen("Hello world!");
* “indexof”

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

* Substring – Start the slice at index 6 and end the slice 5 positions later:

$x = "Hello World!";

echo substr($x, 6, 5);

* Substring – Start the slice at index 6 and go all the way to the end:

$x = "Hello World!";

echo substr($x, 6);

* String reference: <https://www.w3schools.com/php/php_ref_string.asp>
	1. Numbers
* There are three main numeric types in PHP:
* Integer
* Float
* Number Strings

In addition, PHP has two more data types used for numbers:

* Infinity
* NaN
* Constants – PHP has the following predefined constants for integers:
* PHP\_INT\_MAX - The largest integer supported
* PHP\_INT\_MIN - The smallest integer supported
* PHP\_INT\_SIZE -  The size of an integer in bytes

PHP has the following predefined constants for floats (from PHP 7.2):

* PHP\_FLOAT\_MAX - The largest representable floating point number
* PHP\_FLOAT\_MIN - The smallest representable positive floating point number
* PHP\_FLOAT\_DIG - The number of decimal digits that can be rounded into a float and back without precision loss
* PHP\_FLOAT\_EPSILON - The smallest representable positive number x, so that x + 1.0 != 1.0
* Check for int or float, or infinity – PHP has the following functions to check if the type of a variable is integer:
* is\_int()
* is\_integer() - alias of is\_int()
* is\_long() - alias of is\_int()

PHP has the following functions to check if the type of a variable is float:

* is\_float()
* is\_double() - alias of is\_float()

PHP has the following functions to check if a numeric value is finite or infinite:

* [is\_finite()](https://www.w3schools.com/php/func_math_is_finite.asp)
* [is\_infinite()](https://www.w3schools.com/php/func_math_is_infinite.asp)

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)
* Cast float and string to integer:

// 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;

* 1. Misc
* Math: <https://www.w3schools.com/php/php_math.asp>
* Magic Constants: <https://www.w3schools.com/php/php_magic_constants.asp>
* Operators: <https://www.w3schools.com/php/php_operators.asp>
* if/else: <https://www.w3schools.com/php/php_if_else.asp>
* Loops: <https://www.w3schools.com/php/php_looping.asp>
* Functions: <https://www.w3schools.com/php/php_functions.asp>
* Arrays: <https://www.w3schools.com/php/php_arrays.asp>
* Superglobals: <https://www.w3schools.com/php/php_superglobals.asp>