

Introduction to PHP – Part 1

Primary source: <https://www.w3schools.com/php/>

Internet Programming 2, Lesson 1

April 2024

About PHP
Basic PHP Syntax
Comments in PHP
PHP Case Sensitivity
PHP Variables & Scope
PHP Constants
echo and print

Monday, 29 April 2024

Online, MS Teams, commence at 11h30:

Topics:

1. Announcements: Meta AI + Work Schedule (BB) + Delay in results release
2. PHP Pre-reqs: www Resources Vs IP2 requirements (VS Code without AI + WAMP)
3. Review: "How the www works"
4. Introduction to PHP: PHP Lesson Decks + Quiz + Tutorials

PK Ramdeyal

Although you merely require the PHP tutorial from the W3schools site as a reference (textbook) for the structured programming concepts in PHP that we will cover in IP2, here are a few resources that you may find useful for engaging with the PHP language, going forward:

<http://phptester.net/>

<https://php-fiddle.com/>

<https://phptherightway.com/>

<https://replit.com/ai>

<https://onlinephp.io/>

<https://www.php.net/docs.php>

The last one in the above list is extremely useful for PHP documentation and reference materials.

What is PHP?

PHP: **H**ypertext **P**re-processor

widely-used, open source scripting language

Server-side scripting language

Free, open-source

Popular language!

biggest blogging system WordPress

largest social network (Facebook)!

Easy to learn and use

Recall

- HTML mark-up (source code) are stored in text files with the .htm or .html filename extension
- HTML files contain CSS source code, but if we wish to use external CSS, these are stored in text files but with a .css filename extension
- HTML files also contain JavaScript source code, but if we wish to use an external JavaScript file, then we name this with a .js filename extension

So what are PHP files?

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

What Can PHP Do?

- generate **dynamic** page content
- create, open, read, write, delete, and close **files** on the server
- **collect form data**
- send and receive **cookies**
- add, delete, modify data in your **database**
- **control user-access**
- **encrypt data**

output images, PDF files, and even Flash movies; any text, such as XHTML and XML.

Why PHP?

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

~~Version: PHP 7~~

- ~~• PHP 7 is much faster than the previous popular stable release (PHP 5.6)~~
- ~~• PHP 7 has improved Error Handling~~
- ~~• PHP 7 supports stricter Type Declarations for function arguments~~
- ~~• PHP 7 supports new operators (like the spaceship operator: \lessseqgtr)~~

Version: PHP 8

- PHP Version: from PHP 7 to the latest PHP 8.x version.
- Have a look at the last url provided earlier for the latest updates on PHP versions and support therefor.
- Appropriate screen shot from https://www.php.net/releases/8_3_2.php below:

PHP 8.3.2 Release Announcement

The PHP development team announces the immediate availability of PHP 8.3.2. This is a bug fix release.

All PHP 8.3 users are encouraged to upgrade to this version.

For source downloads of PHP 8.3.2 please visit our [downloads page](#), Windows source and binaries can be found on windows.php.net/download/. The list of changes is recorded in the [ChangeLog](#).

MS Copilot Response, 30 April 2024

The major differences between PHP 7 and PHP 8 include:

- Performance: PHP 8 introduces the Just-In-Time (JIT) compiler which can significantly improve performance for certain types of workloads³.
- Syntax: PHP 8 requires less code for the same tasks due to better syntax¹.
- Type Safety: Improved type safety with more robust typing support¹.
- Named Arguments: This feature allows developers to pass arguments based on their names rather than order, which is especially useful for functions with many optional parameters¹.
- Attributes: Also known as annotations in other languages, attributes in PHP 8 provide a way to add metadata to classes, methods, and properties³.
- Match Expression: A new match expression is introduced in PHP 8, which is a more powerful and concise alternative to the switch statement³.
- Constructor Property Promotion: This feature reduces boilerplate code by allowing the declaration of class properties and their assignment directly in the constructor¹.

These improvements make PHP 8 faster, more efficient, and more user-friendly for developers compared to PHP 7. For a more detailed comparison, you might want to check out the resources available online¹²³.

Source: Conversation with Bing, 2024/04/30

(1) PHP 7 vs. PHP 8: Unveiling Key Differences and Powerful Feature. <https://vickyhhetri.com/2023/11/09/some-key-differences-between-php-7-and-php-8/>.

(2) PHP 7 vs PHP 8: A Comparison | Liquid Web. <https://www.liquidweb.com/kb/php-7-vs-php-8/>.

(3) PHP 7 vs PHP 8 | Top 6 Differences You Should Know - EDUCBA. <https://www.educba.com/php-7-vs-php-8/>.

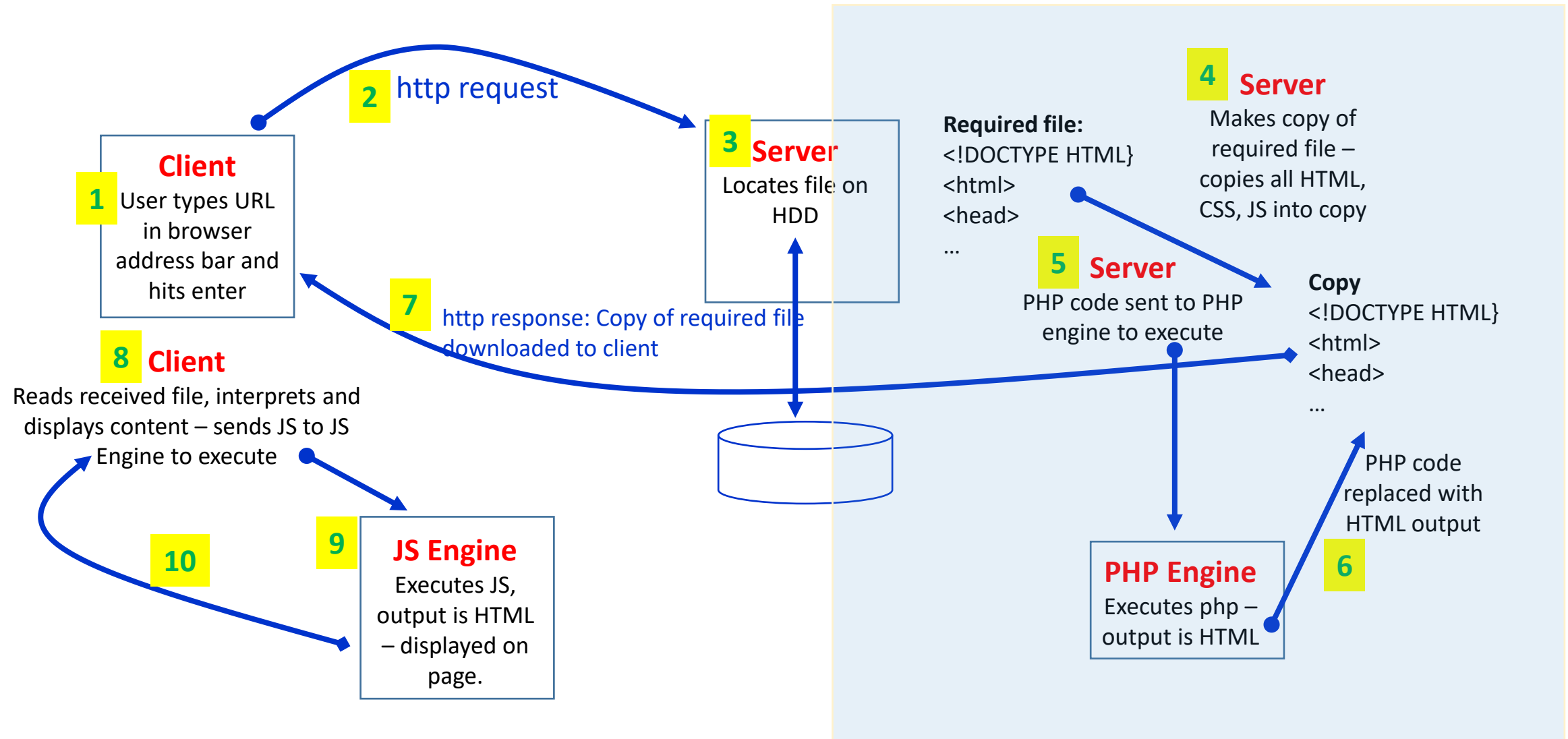
(4) PHP 7 vs PHP 8 | Key Differences and Comparison Table. <https://www.ameerpet.org/php-7-vs-php-8.html>.

(5) PHP 8 Infographic: What's New vs PHP 7 - Super Dev Resources. <https://superdevresources.com/php-8-infographic/>.

(6) en.wikipedia.org. <https://en.wikipedia.org/wiki/PHP>.

Introduction – “The bigger picture”

Recall the *conceptual view* of how the web works



Locally Simulated Server Environment - WAMP

We will use the **WAMP** package to simulate a local server (localhost) to execute our PHP scripts, and **Notepad+** to create and edit out PHP scripts. Source files will be stored in a solutions folder (remember to name your folder using only a single word – no spaces) in the **www folder** contained in the WAMP folder on the C: drive.

W-Windows

A-Apache

M-MySQL

P- PHP

This is an environment in which a server is simulated in the local machine – mainly, by sharing resources and maintaining strict channels between the client (local browser) and the simulated server...used for development, but can be used for production purposes

Development cycle using WAMP and Notepad+

- Run WAMP
- Create sub-folder in **www** folder (which you should find in the WAMP dir)
- Create PHP source file in notepad+ (or VS Code) and save in above folder [.php]
- Open localhost [WAMP]
- Navigate to php source file and run (in localhost)
- Localhost IP is: **127.0. 0.1** (the loopback address)

Basic PHP Syntax

A PHP script can be placed anywhere in the document. For now we keep things simple and write php in the body section.

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

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

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

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello PHP!" on a web page:

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

Output

My first PHP page

Hello PHP!

Note: PHP statements end with a semicolon (;).

Since the output from the php script is html, we can include any html tags, including CSS in the echo statements.

Errors during Development

Here is an example in which we make a simple error to review how to deal with errors during development.

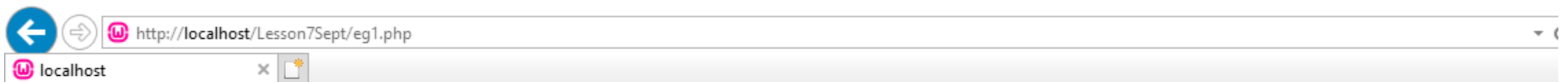
```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo1 "Hello World!";
?>

</body>
</html>
```

An attempt to run this code results in the following error:



(!) Parse error: syntax error, unexpected "'Hello World!'" (T_CONSTANT_ENCAPSED_STRING) in C:\wamp64\www\Lesson7Sept\eg1.php on line 8

EXPLAIN:

```
<?php
echo "Welcome to PHP Server side scripting <br>";
// echo"\n";
echo "My first PHP script!";
?>
```

It is important to keep in mind that the output from the PHP script goes to the file the server is currently preparing to send back to the user/client browser. In this regard, HTML, CSS and JS can be included as part of the echo parameter. Moreover, any additional “formatting” of the source code in the copied file can also be included in the file. As an example, including of the br-element in the code above, means the developer wanted a new line on the web page itself – in this case a
 element will be added to the code and the client browser can then interpret this html. On the other hand, ‘echo “\n”;' inserts a newline character in the source file itself. You will recall that the backslash is used to indicate an escape code, a newline character in this case. Look at the source code in the browser, with and without the ‘echo “\n”;' to see the difference.

Comments in PHP

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- *Let others understand what you are doing*
- *Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code*

PHP supports several ways of commenting:

```
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
/   This is a multiple-lines comment block
    that spans over multiple lines. The comment-block may also span a few
    characters on a single line as in the example below/

// You can also use comments to leave out parts of a code line
$x = 5 / + 15 / + 5;
echo $x;
?>
</body>
</html>
```

Output

10

PHP Case Sensitivity

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are **NOT case-sensitive**. *However we will stick to lowercase for these and camelCase for user-defined identifiers. In the example, we will use the second option!*

In the example below, all three echo statements below are legal (and equal):

```
<!DOCTYPE html>
<html>
<body>
<?php
    ECHO "Hello World!<br>";
    echo "Hello World!<br>";
    EcHo "Hello World!<br>";
?>

</body>
</html>
```

Output

```
Hello World!
Hello World!
Hello World!
```

However; all variable names are case-sensitive.

In the example below, only the first statement will display the value of the \$color variable (this is because \$color, \$COLOR, and \$coLOR are treated as three different variables): we will use the first option – lower camel case!

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

<?php
    $color = "red";
    echo "My car is " . $color . "<br>";
    echo "My house is " . $COLOR . "<br>";
    echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Output

My car is red
My house is
My boat is

Creating (Declaring) PHP Variables

Variables are "containers" for storing information.

In PHP, a variable starts with the \$ sign, followed by the name of the variable:

<?php

```
$txt = "Hello world!";  
$x = 5;  
$y = 10.5;  
echo $txt;  
echo "<br>";  
echo $x;  
echo "<br>";  
echo $y;
```

?>

Output

```
Hello world!  
5  
10.5
```

Note: Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Think of variables as containers for storing data. *You will no doubt recall that variable names are merely "symbolic names for memory addresses/locations!!!yes?"*

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

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

Remember that PHP variable names are case-sensitive!

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

The following two examples will produce the same output: note the use of the dot as a string concatenation command

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

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

Output

I love W3Schools.com!

The following example will output the sum of two variables:

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

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

Output

9

```
</body>
</html>
```

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on its value. In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

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.

You will learn more about strict and non-strict requirements, and data type declarations in the PHP Functions chapter.

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared **outside** a function has a GLOBAL SCOPE and **can only** be accessed outside a function: (unlike many other programming languages)

```
<body>
<?php
    $x = 5; // global scope
    function myTest() {
        // using x inside this function will generate an error
        echo "<p>Variable x inside function is: $x</p>";
    }
    myTest();
    echo "<p>Variable x outside function is: $x</p>";
?>
</body>
```

Output

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

Global and Local Scope

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

```
1 <body>
2 <?php
3     function myTest() {
4         $x = 5; // local scope
5         echo "<p>Variable x inside function is: $x</p>";
6     }
7     myTest();
8
9     // using x outside the function will generate an error
10    echo "<p>Variable x outside function is: $x</p>";
11    ?>
12 </body>
```

Output

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

You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.

PHP The global Keyword

Recall that a global variable cannot ordinarily be accessed from within a function in PHP. But we can use the global keyword to access a global variable from within a function. To do this, use the global keyword before the variables (inside the function), "as if you were declaring them":

```
1  <?php
2  $x = 5;
3  $y = 10;
4
5  function myTest() {
6      global $x, $y;
7      $y = $x + $y;
8  }
9
10 myTest(); // run function
11 echo $y; // output the new value for variable 12 $y
13 ?>
```

Output

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:

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

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y;
?>
```

Output

15

PHP The static Keyword

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:

```
1  <?php
2  function myTest() {
3      static $x = 0;
4      echo $x;
5      $x++;
6  }
7
8  myTest();
9  echo "<br>";
10 myTest();
11 echo "<br>";
12 myTest();
13 ?>
```

Output

0

1

2

Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.

Note: The variable is still local to the function.

PHP Constants

Constants are like variables except that once they are defined they cannot be changed or undefined.

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)
```

Parameters:

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

The example below creates a constant with a **case-sensitive** name:


```
<?php
// case-sensitive constant name
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

Output

Welcome to W3Schools.com!

The example below creates a constant with a **case-insensitive** name:

```
<?php
// case-insensitive constant name
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

Output

Welcome to W3Schools.com!

Constants are Global

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

The example below uses a constant inside a function, even if it is defined outside the function:

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

function myTest() {
    echo GREETING;
}
myTest();
?>
```

Output

Welcome to W3Schools.com!

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.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with
multiple parameters.";
?>
```

Output

PHP is Fun!

Hello world!

I'm about to learn PHP!

This string was made with multiple parameters.

Display Variables

The following example shows how to output text and variables with the echo statement:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

Output

Learn PHP

Study PHP at W3Schools.com

9

The PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

Output

PHP is Fun!
Hello world!
I'm about to learn PHP!

Display Variables

The following example shows how to output text and variables with the print statement:

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

Output

Learn PHP
Study PHP at W3Schools.com
9

Exercise 1:

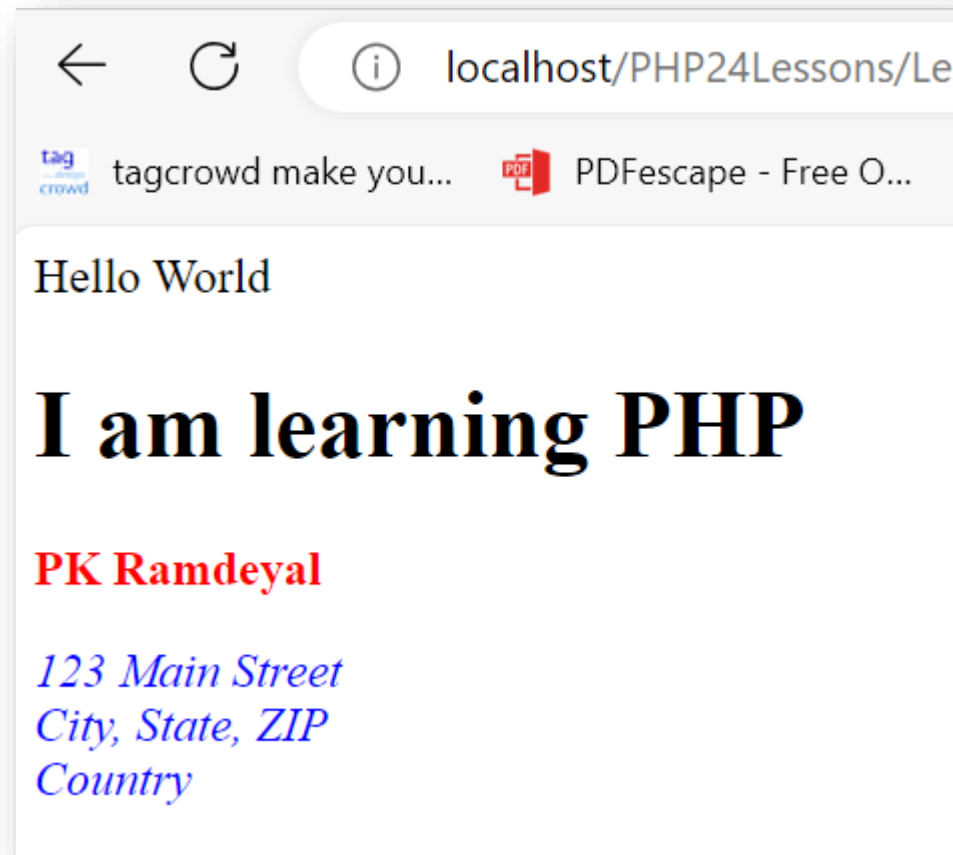
1-1. Create a new PHP file (call this file Ex1-1.php) and save this file in the folder you created within the www folder.

Now write php code to do the following:

- o Display the text Hello World in default text;
- o Display the text “I am learning PHP” in a heading-1 element;
- o Display your name to appear in red text and in bold face font;
- o Display your address (4-lines) below your name in blue text and in italics.

1-2. Describe the difference(s) between the PHP echo and print statements. Display your answer in a table on a web page using the PHP echo statement. Save your source code as Ex1-2.php

```
<?php
    echo "Hello World";
    echo "<h1>I am learning PHP</h1>";
    echo "<p style='color: red; font-weight: bold;'>PK Ramdeyal</p>";
    echo "<p style='color: blue; font-style: italic;'>123 Main Street<br>City, State,
ZIP<br>Country</p>";
?>
```



AI Chatbot Response:

The ``echo`` and ``print`` statements in PHP are both used to output strings. However, there are some differences between them:

1. Outputting Multiple Values:

- ``echo``: Can output multiple values separated by commas.
- ``print``: Outputs a single value. It cannot print multiple values directly.

2. Return Value:

- ``echo``: Does not return a value. It's a language construct.
- ``print``: Returns 1, indicating success. It's a function.

3. Usage:

- ``echo``: Is a language construct, not a function, so the parentheses are optional.
- ``print``: Is a function, so parentheses are mandatory.

4. Performance:

- ``echo``: Is slightly faster than ``print``.
- ``print``: Is slightly slower than ``echo``.

In practical terms, the choice between ``echo`` and ``print`` often comes down to personal preference and specific use cases. ``echo`` tends to be more commonly used due to its slightly better performance and more flexible syntax, especially when outputting HTML or complex strings. On the other hand, ``print`` can be useful when you need to explicitly capture its return value for error checking or flow control.