



**Institut
supérieur
d'informatique**

Introduction to PHP Web1909A



Amirhossein Ghasemi

May 2020

255 Crémazie Est, bureau 100, Montréal (Québec) H2M 1M2
Téléphone : (514) 842-2426, télécopieur : (514) 842-2084
www.isi-mtl.com

PHP Introduction

- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- PHP is a widely used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP 7 is the latest stable release.
- PHP code is executed on the server.
- PHP was created in 1994 by Rasmus Lerdof

What You Should Already Know

Before you continue you should have a basic understanding of the following:

- HTML
- CSS
- JavaScript

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use
- PHP is an amazing and popular language!

- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)!
- It is also easy enough to be a beginner's first server-side language!

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"**

What Can PHP Do?

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

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

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

PHP Installation

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support

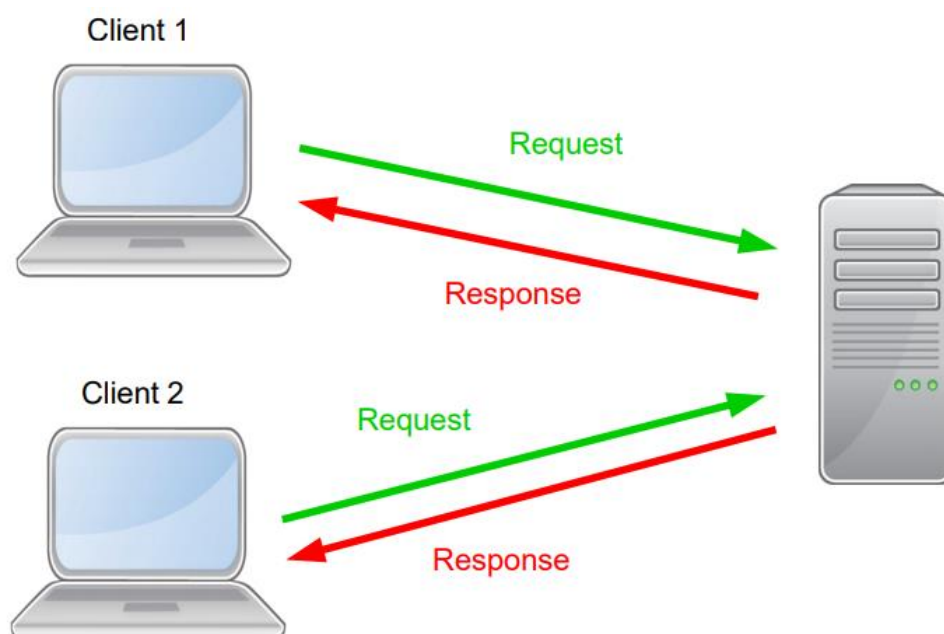
- Install a web server (XAMPP) on your own PC, and then install PHP and MySQL
- If your server has activated support for PHP, you do not need to do anything.
- Just create some .php files, place them in your web directory (C:\xampp\htdocs), and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.
- Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

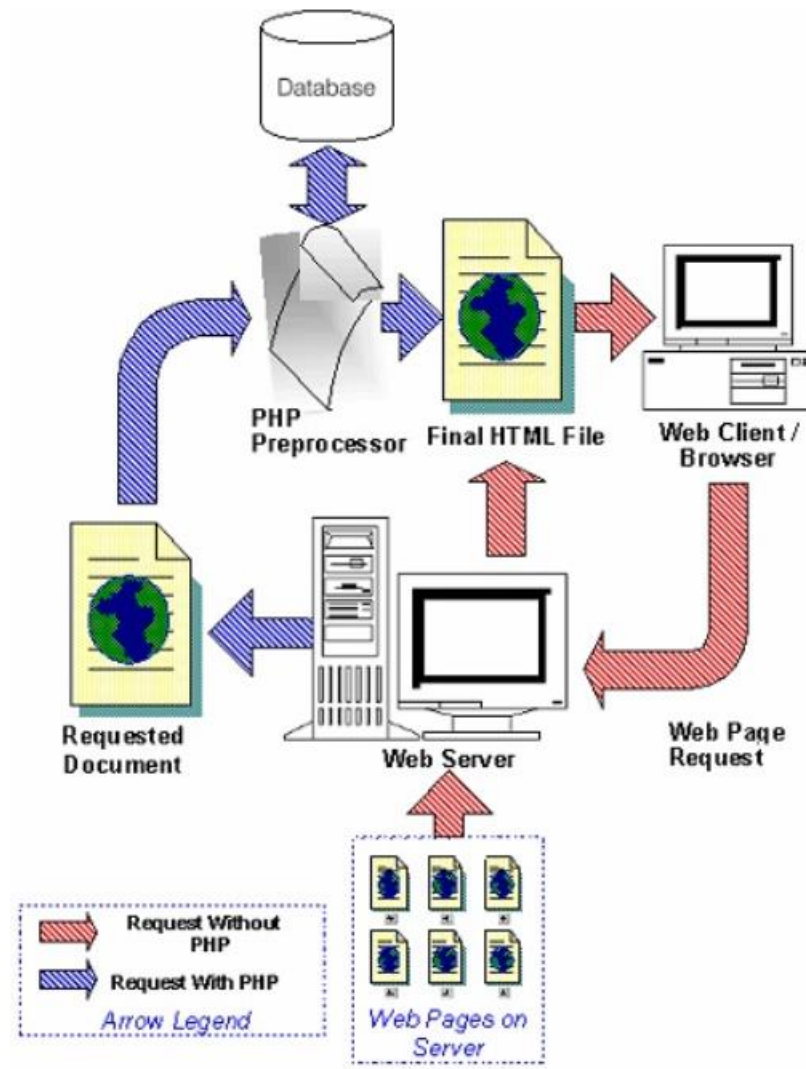
However, if your server does not support PHP, you must:

- install a web server (XAMPP)

HTTP Protocol



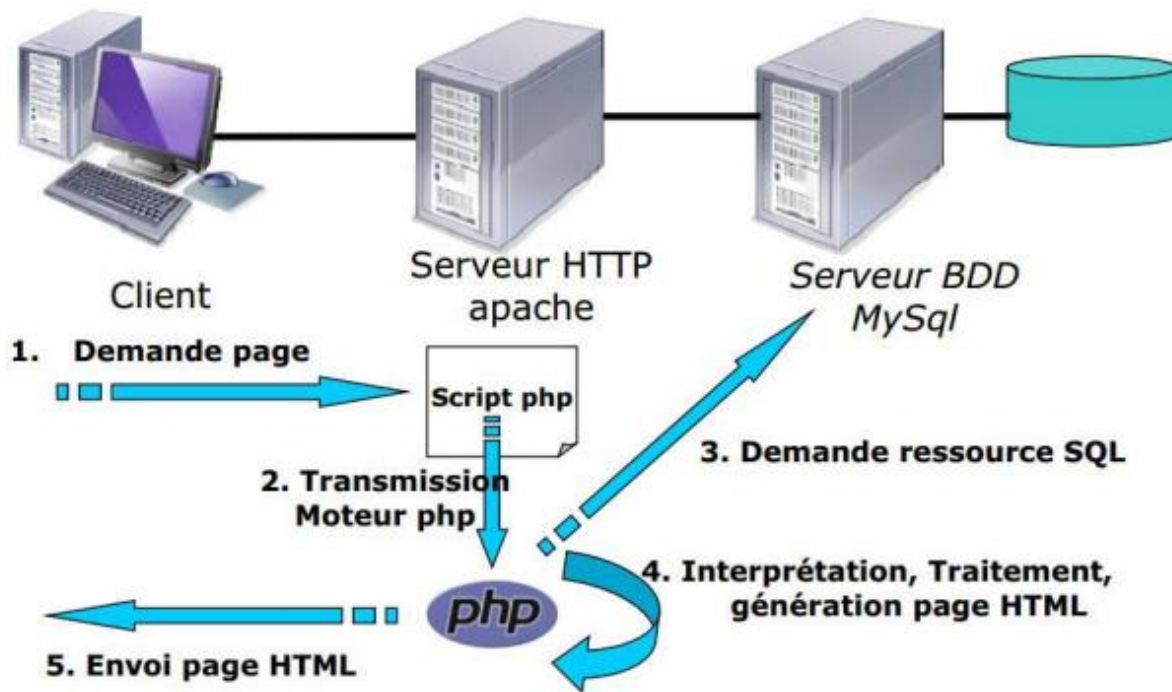
HTTP Details



(From "An Introduction to PHP" by John Coggeshallm)

Http with Apache/PHP

- Static websites: HTML et CSS.
- Dynamic websites: PHP, Java, Ruby



PHP Tags

- `<?php ?>`

Allows you to insert PHP code in the middle of HTML.

(E1):

```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First PHP Page</h1>
    <?php
      echo 'Hello World!';
    ?>
  </body>
</html>
```

My First PHP Page

Hello World!

- `<?= ?>`

Insert text interpreted by PHP in HTML. These two code segments produce the same result. (E2)

```
1 <?php
2     $nom = "Hello World!";
3     ?>
4 <!DOCTYPE html>
5 <html>
6     <body>
7         <h1>My First PHP Page</h1>
8         <?= $nom ?>
9     </body>
10 </html>
```

- You can use PHP tags anywhere in a page.
- The file is saved with the .php extension.

PHP Variables

- PHP is a weakly typed language
- To declare a variable, the name must start with a \$
- Ex:
 - \$nom
 - \$age
 - \$isPresent or \$is_present
- Special characters like +, -, %, (,), &, etc are not allowed

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
- 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)

PHP Best Practices

- Use the <?php and ?> tag in php and html files
- Use only the opening tag <?php in files only containing php
- To be continued in the course: cross site scripting, data verification, using includes

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

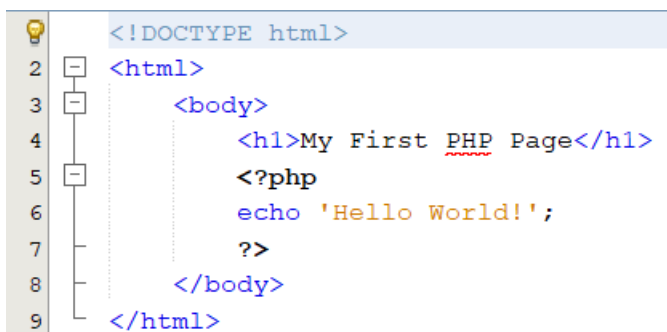
- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:

<?php

// PHP code goes here

?>

- 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 World!" on a web page:



```
<!DOCTYPE html>
<html>
  <body>
    <h1>My First PHP Page</h1>
    <?php
      echo 'Hello World!';
    ?>
  </body>
</html>
```

Output

My first PHP page

Hello World!

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

PHP Case Sensitivity

- In PHP, NO keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are case-sensitive.
- In the example below, all three echo statements below are equal and legal (E3):



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

Output

localhost

Hello World!
Hello World!
Hello World!

Note: However; all variable names are case-sensitive!

Look at 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** (E4):

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <?php
5       $color = "red";
6       echo "My car is " . $color . "<br>";
7       echo "My house is " . $COLOR . "<br>";
8       echo "My boat is " . $coLOR . "<br>";
9     ?>
10  </body>
11 </html>

```

Output

My car is red

Notice: Undefined variable: COLOR in C:\xampp\htdocs\Day1\E4.php on line 7
My house is

Notice: Undefined variable: coLOR in C:\xampp\htdocs\Day1\E4.php on line 8
My boat is

PHP Comments

Comments in PHP

A comment in PHP code is a line that is not executed as a 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 your code
- 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:

Example

Syntax for single-line comments (E5):

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <?php
5       // This is a single-line comment
6
7       # This is also a single-line comment
8     ?>
9   </body>
10 </html>

```

Example (E6):

Syntax for multiple-line comments:

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <?php
5       /*
6       This is a multiple-lines comment block
7       that spans over multiple
8       lines
9       */
10    ?>
11  </body>
12 </html>

```

Example

Using comments to leave out parts of the code (E7):

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4     <?php
5       // You can also use comments to leave out parts of a code line
6       $x = 5 /* + 15 */ + 5;
7       echo $x;
8     ?>
9   </body>
10 </html>

```

Creating (Declaring) PHP Variables

In PHP, a variable starts with the \$ sign, followed by the name of the variable (E8):

Example

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       $txt = "Hello world!";
7       $x = 5;
8       $y = 10.5;
9
10      echo $txt;
11      echo "<br>";
12      echo $x;
13      echo "<br>";
14      echo $y;
15    ?>
16
17  </body>
18 </html>
```

OUTPUT

```
Hello world!
5
10.5
```

After the execution of the statements above, the variable \$txt will hold the value Hello world!, the variable \$x will hold the value 5, and the variable \$y will hold the value 10.5.

Note: When you assign a text value to a variable, put quotes around the value.

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.

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 (E9):

Example

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       $txt = "W3Schools.com";
7       echo "I love $txt!";
8     ?>
9
10  </body>
11 </html>
```

OUTPUT

I love W3Schools.com!

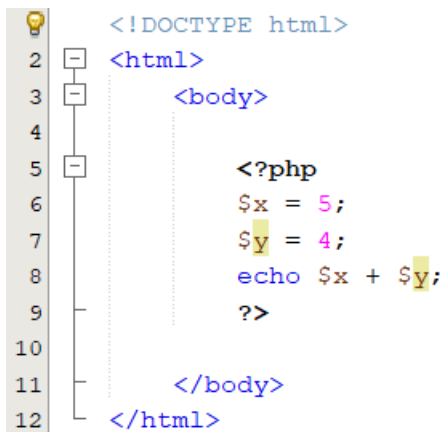
The following example will produce the same output as the example above (E10):

Example

```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       $txt = "W3Schools.com";
7       echo "I love " . $txt . "!";
8     ?>
9
10  </body>
11 </html>
```

OUTPUT

The following example will output the sum of two variables (E11):

A screenshot of a code editor with a light blue background. On the left, there is a vertical line with line numbers from 2 to 12. To the right of the line numbers is the code. The code is a PHP script embedded in an HTML document. It starts with a DOCTYPE declaration, followed by an HTML tag, a body tag, and a PHP block. Inside the PHP block, two variables, \$x and \$y, are assigned the values 5 and 4 respectively. Then, the sum of \$x and \$y is calculated and output using the echo statement. The code ends with the closing PHP tag, body tag, and HTML tag. The variables \$x and \$y, and the values 5 and 4, are highlighted in yellow in the original image.

```
<!DOCTYPE html>
<html>
<body>
<?php
    $x = 5;
    $y = 4;
    echo $x + $y;
?>
</body>
</html>
```

OUTPUT

9

Note: You will learn more about the echo statement and how to output data to the screen in the next chapter.

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 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.

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:

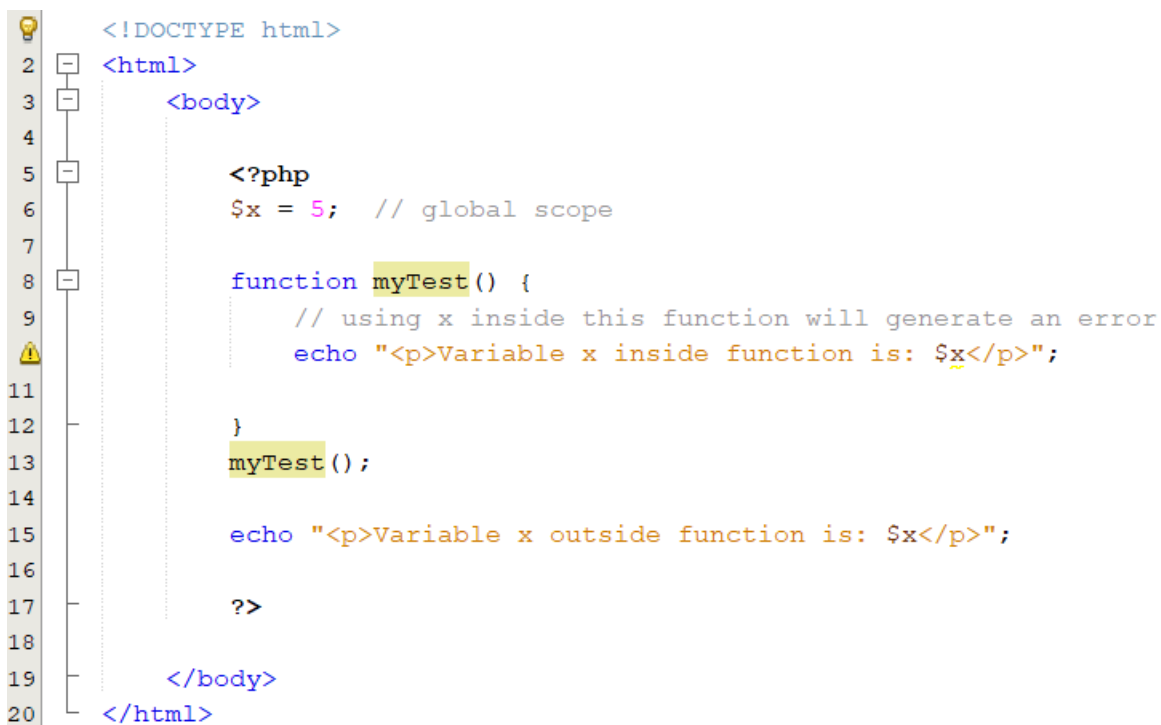
- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

Example

Variable with global scope (E12):



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

    <?php
      $x = 5; // global scope

      function myTest() {
        // using x inside this function will generate an error
        echo "<p>Variable x inside function is: $x</p>";
      }
      myTest();

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

    ?>

  </body>
</html>
```

Output

Notice: Undefined variable: x in **C:\xampp\htdocs\Day1\E12.php** on line **10**

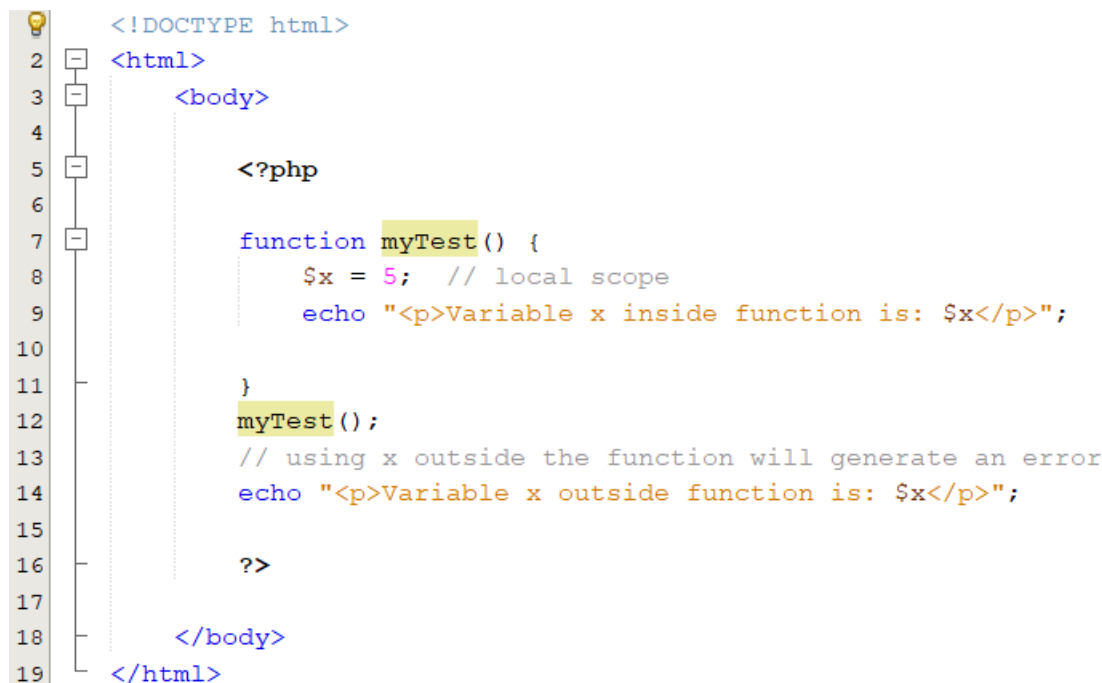
Variable x inside function is:

Variable x outside function is: 5

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function (E13):

Example

Variable with local scope:



```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6
7     function myTest() {
8         $x = 5; // local scope
9         echo "<p>Variable x inside function is: $x</p>";
10
11     }
12     myTest();
13     // using x outside the function will generate an error
14     echo "<p>Variable x outside function is: $x</p>";
15
16     ?>
17
18   </body>
19 </html>
```

OUTPUT

Variable x inside function is: 5

Notice: Undefined variable: x in **C:\xampp\htdocs\Day1\E13.php** on line **14**

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

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) (E14):

Example

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

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

    function myTest() {
        global $x, $y;
        $y = $x + $y;
    }

    myTest(); // run function
    echo $y; // output the new value for variable $y
?>

</body>
</html>
```

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 (E15):

Example

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       $x = 5;
7       $y = 10;
8
9       function myTest() {
10         $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
11
12       }
13
14       myTest();
15       echo $y;
16
17     ?>
18
19   </body>
20 </html>

```

OUTPUT

15

PHP The static Keyword

Normally, when a function is completed/executed, all 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 (E16):

Example

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       function myTest() {
7         static $x = 0;
8         echo $x;
9         $x++;
10
11       }
12
13       myTest();
14       echo "<br>";
15       myTest();
16       echo "<br>";
17       myTest();
18
19     ?>
20   </body>
21 </html>

```

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 echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.

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.

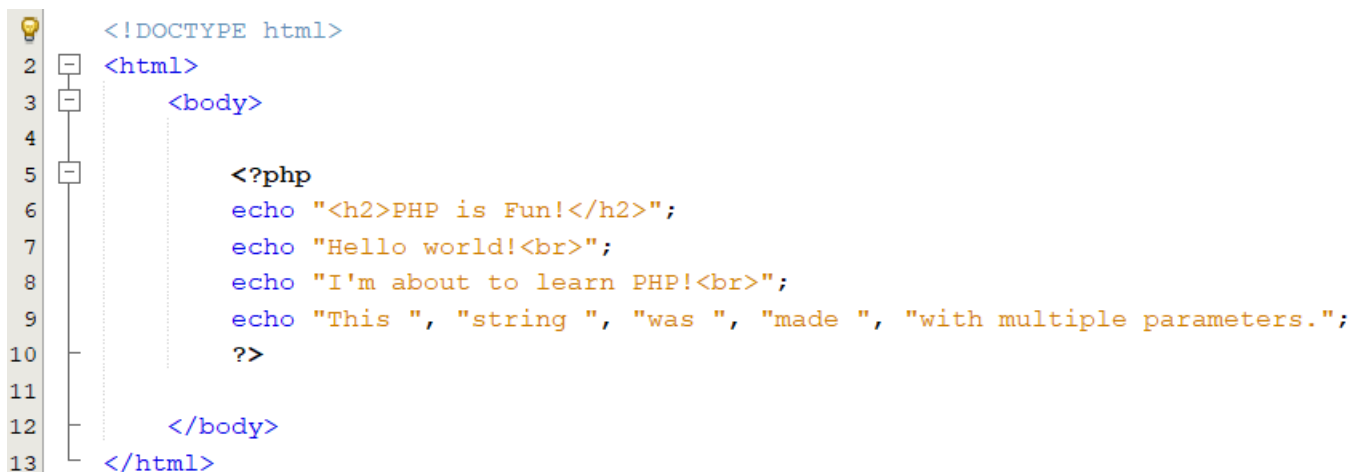
The PHP echo Statement

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) (E17):

Example:



```
1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       echo "<h2>PHP is Fun!</h2>";
7       echo "Hello world!<br>";
8       echo "I'm about to learn PHP!<br>";
9       echo "This ", "string ", "was ", "made ", "with multiple parameters.";
10      ?>
11
12   </body>
13 </html>
```

OUTPUT

PHP is Fun!

Hello world!

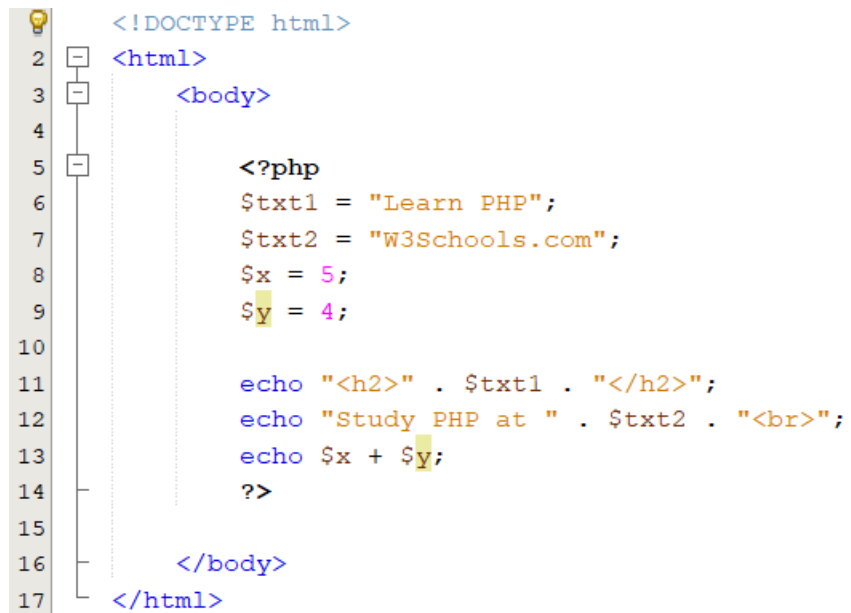
I'm about to learn PHP!

This string was made with multiple parameters.

Display Variables

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

Example (E18):

A screenshot of a code editor with a light blue background. On the left, there is a vertical line with numbered markers from 1 to 17. The code is written in a syntax-highlighted format. It starts with a light blue icon at line 1. The code is as follows:

```
<!DOCTYPE html>
<html>
  <body>
    <?php
      $txt1 = "Learn PHP";
      $txt2 = "W3Schools.com";
      $x = 5;
      $y = 4;

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

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) (E19):

Example:

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       print "<h2>PHP is Fun!</h2>";
7       print "Hello world!<br>";
8       print "I'm about to learn PHP!";
9     ?>
10
11   </body>
12 </html>

```

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 (E20):

Example:

```

1 <!DOCTYPE html>
2 <html>
3   <body>
4
5     <?php
6       $txt1 = "Learn PHP";
7       $txt2 = "W3Schools.com";
8       $x = 5;
9       $y = 4;
10
11       print "<h2>" . $txt1 . "</h2>";
12       print "Study PHP at " . $txt2 . "<br>";
13       print $x + $y;
14     ?>
15
16   </body>
17 </html>

```

OUTPUT

Learn PHP

Study PHP at [W3Schools.com](https://www.w3schools.com)

9