COURSE: PHP fundamentals

SLIDE ID: slide 2 tutorial

TUTOR: Ibrahim U Kamara

PHP VARIABLES

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

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

Example

```
<?php

$txt = "Hello world!";

$x = 5;

$y = 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.

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

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:

Example

```
<?php

$txt = "W3Schools.com";

echo "I love $txt!";

?>
```

The following example will produce the same output as the example above:

Example

```
<?php

$txt = "W3Schools.com";

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

?>
```

The following example will output the sum of two variables:

Example

```
<?php

$x = 5;

$y = 4;

echo $x + $y;

?>
```

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 DATA TYPES

Variables can store data of different types, and different data types can do different things

PHP supports the following data types

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

PHP Strings

A string is a sequence of characters, like "Hello world!".

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

strlen() - Return the Length of a String

The PHP strlen() function returns the length of a string.

Example

Return the length of the string "Hello world!":

```
<?php

echo strlen("Hello world!"); // outputs 12

?>
```

str_word_count() - Count Words in a String

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php

echo str_word_count("Hello world!"); // outputs 2

?>
```

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

?>
```

strpos() - Search For a Text Within a String

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return

FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php

echo strpos("Hello world!", "world"); // outputs 6

?>
```

Tip: The first character position in a string is 0 (not 1).

str_replace() - Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php

echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!

?>
```

Note: I gave you guys assignment on data type right? So, the remaining data types Will be discuss during the class as assessment and class work, so be ready.

PHP Math

PHP has a set of math functions that allows you to perform mathematical tasks on numbers.

PHP pi() Function

The pi() function returns the value of PI:

Example

```
<?php
```

```
echo(pi()); // returns 3.1415926535898

?>
```

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 abs() Function

The abs() function returns the absolute (positive) value of a number:

Example

```
<?php

echo(abs(-6.7));  // returns 6.7

?>
```

PHP sqrt() Function

The sqrt() function returns the square root of a number:

Example

```
<?php

echo(sqrt(64));  // returns 8

?>
```

PHP round() Function

The round() function rounds a floating-point number to its nearest integer:

Example

```
<?php
echo(round(0.60));  // returns 1
echo(round(0.49));  // returns 0
?>
```

Random Numbers

The rand() function generates a random number:

Example

```
<?php
echo(rand());
?>
```

PHP IF...ELSE...ELSEIF STATEMENTS

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- **if statement -** executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif...else statement -** executes different codes for more than two conditions

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
```

```
  code to be executed if condition is true;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
}
?>
```

PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

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

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php
$t = date("H");
if ($t < "20") {
```

```
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

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

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
```

```
  echo "Have a good night!";
}
?>
```

PHP Loops

In the following chapters you will learn how to repeat code by using loops in PHP.

PHP LOOPS

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

PHP WHILE LOOP

The while loop - Loops through a block of code as long as the specified condition is true.

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

Examples

The example below displays the numbers from 1 to 5:

Example

```
<?php
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Example Explained

$x = 1; - Initialize the loop counter ($x), and set the start value to 1

$x <= 5 - Continue the loop as long as $x is less than or equal to 5

$x++; - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
$x = 0;
while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 100 - Continue the loop as long as $x is less than or equal to 100

$x+=10; - Increase the loop counter value by 10 for each iteration

PHP DO WHILE LOOP

The do...while loop - Loops through a block of code once, and then repeats the loop as long as the specified condition is true.

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

Examples

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the

condition is false. See example below.

This example sets the $x variable to 6, then it runs the loop, and then the condition is checked:

Example

```
<?php
$x = 6;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

PHP FOR LOOP

The for loop - Loops through a block of code a specified number of times.

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it

evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 10; - Continue the loop as long as $x is less than or equal to 10

$x++ - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 100; - Continue the loop as long as $x is less than or equal to 100

$x+=10 - Increase the loop counter value by 10 for each iteration

PHP FOREACH LOOP

The foreach loop - Loops through a block of code for each element in an array.

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

```
foreach ($array as $value) {
  code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

The following example will output the values of the given array ($colors):

ExampleGet your own PHP Server

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

The following example will output both the keys and the values of the given array ($age):

Example

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

```
foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

PHP BREAK AND CONTINUE

PHP Break

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This example jumps out of the loop when x is equal to 4:

Example

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
?>
```

PHP Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

Example

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
?>
```

BREAK AND CONTINUE IN WHILE LOOP

You can also use break and continue in while loops:

Break Example

```
<?php
$x = 0;

while($x < 10) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Continue Example

```
<?php
```

```
$x = 0;


while($x < 10) {
  if ($x == 4) {
    $x++;
    continue;
  }
  echo "The number is: $x <br>";
  $x++;
}
?>
```

ASSIGNMENT FOR NEXT CLASS

- PHP variable scope
- PHP operators
- PHP switch statements
- PHP constant