PHP Programming

WEEK 6-8

1

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

2

What is a PHP File?

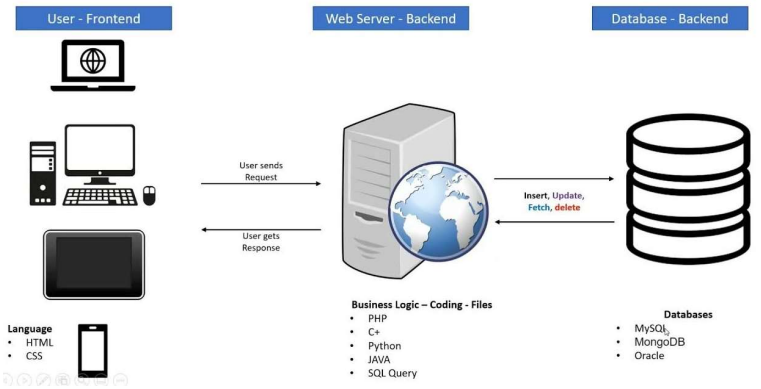
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

3

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

4



5

PHP Installation

6

What Do I Need?

- To start using PHP, you can:
  - Find a web host with PHP and MySQL support
  - Install a web server on your own PC, and then install PHP and MySQL
  - Or (next page)

7

Download and Install

- XAMPP
  - a cross-platform web server that is free and open-source.
  - short form for Cross-Platform, Apache, MySQL, PHP, and Perl.
  - a popular cross-platform web server that allows programmers to write and test their code on a local webserver.
- WAMP
  - an acronym that stands for Windows, Apache, MySQL, and PHP.
  - acts like a virtual server on your computer.

8

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
?>
```

9

- The default file extension for PHP files is ".php".
- A PHP file normally contains HTML tags, and some PHP scripting code.

10

Example

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

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

11

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

12

Example

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

<?php
// This is a single-line comment

# This is also a single-line comment
?>

</body>
</html>
```

13

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;
?>
```

- 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.

14

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!

15

Output Variables

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

```
<?php
      $txt = “TUPVisayas";
      echo "I love $txt!";
?>
```

or

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

16

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

17

Global Scope

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

```php
<?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>";
?>
```

18

Local Scope

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

```php
<?php
	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>";
?>
```

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.

19

PHP echo and print Statements

- With PHP, there are two basic ways to get output: echo and print.
- echo and print are more or less the same. They are both used to output data to the screen.

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

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

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

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

20

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

21

PHP String

- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

```php
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

22

PHP Integer

- An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.
- Rules for integers:
  - An integer must have at least one digit
  - An integer must not have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

23

PHP Integer Example

- In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```php
<?php
$x = 5985;
var_dump($x);
?>
```

24

PHP Float

- A float (floating point number) is a number with a decimal point or a number in exponential form.
- In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

25

PHP Boolean

- A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

- Booleans are often used in conditional testing.

26

PHP Array

- An array stores multiple values in one single variable.
- In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

27

PHP Object

- Classes and objects are the two main aspects of object-oriented programming.
- A class is a template for objects, and an object is an instance of a class.

28

PHP NULL Value

- Null is a special data type which can have only one value: NULL.
- A variable of data type NULL is a variable that has no value assigned to it.
- If a variable is created without a value, it is automatically assigned a value of NULL.
- Variables can also be emptied by setting the value to NULL:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

29

PHP Resource

- The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.
- A common example of using the resource data type is a database call.

30

PHP String Functions

31

strlen() - Return the Length of a String

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

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

32

str_word_count() - Count Words in a String

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

```php
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

33

strrev() - Reverse a String

- The PHP strrev() function reverses a string.

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

34

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.

```php
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

35

str_replace() - Replace Text Within a String

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

```php
<?php
echo str_replace("world", "Dolly", "Hello world!");
// outputs Hello Dolly!
?>
```

36

PHP Numbers

PHP Numbers

- One thing to notice about PHP is that it provides automatic data type conversion.
- So, if you assign an integer value to a variable, the type of that variable will automatically be an integer. Then, if you assign a string to the same variable, the type will change to a string.
- This automatic conversion can sometimes break your code.

PHP Integers

- 2, 256, -256, 10358, -179567 are all integers.
- An integer data type is a non-decimal number between -2147483648 and 2147483647 in 32 bit systems
- and between -9223372036854775808 and 9223372036854775807 in 64 bit systems
- A value greater (or lower) than this, will be stored as float, because it exceeds the limit of an integer

Note: Another important thing to know is that even if 4 * 2.5 is 10, the result is stored as float, because one of the operands is a float (2.5).

PHP Integers

- Here are some rules for integers:
  - An integer must have at least one digit
  - An integer must NOT have a decimal point
  - An integer can be either positive or negative
  - Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

PHP Integers

- 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 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()

41

Example

- Check if the type of a variable is integer:

```
<?php
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
?>
```

42

PHP Floats

- A float is a number with a decimal point or a number in exponential form.
- 2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 are all floats.
- The float data type can commonly store a value up to 1.7976931348623E+308 (platform dependent), and have a maximum precision of 14 digits.

43

PHP Floats

- 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
- PHP has the following functions to check if the type of a variable is float:
  - is_float()
  - is_double() - alias of is_float()

44

Example

- Check if the type of a variable is float:

```php
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

45

PHP Infinity

- A numeric value that is larger than PHP_FLOAT_MAX is considered infinite.
- PHP has the following functions to check if a numeric value is finite or infinite:
  - is_finite()
  - is_infinite()
- However, the PHP var_dump() function returns the data type and value:

```php
<?php
$x = 1.9e411;
var_dump($x);
?>
```

46

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()

```php
<?php
$x = acos(8);
var_dump($x);
?>
```

47

PHP Numerical Strings

- The PHP is_numeric() function can be used to find whether a variable is numeric.
- The function returns true if the variable is a number or a numeric string, false otherwise.

```php
<?php
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));

$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
?>
```

48

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.

```php
<?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;
?>
```

49

PHP Math

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

50

PHP pi() Function

- The pi() function returns the value of PI:

```php
<?php
echo(pi()); // returns 3.1415926535898
?>
```

51

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:

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

52

PHP abs() Function

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

```php
<?php
echo(abs(-6.7));  // returns 6.7
?>
```

53

PHP sqrt() Function

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

```php
<?php
echo(sqrt(64));  // returns 8
?>
```

54

PHP round() Function

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

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

55

Random Numbers

- The rand() function generates a random number:

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

- To get more control over the random number, you can add the optional min and max parameters to specify the lowest integer and the highest integer to be returned.

```php
<?php
echo(rand(10, 100));
?>
```

56

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.

57

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

58

Example

- Create a constant with a case-sensitive name:

```
<?php
define("GREETING", "Welcome to TUPVisayas!");
echo GREETING;
?>
```

- Create a constant with a case-insensitive name:

```
<?php
define("GREETING", "Welcome to TUPVisayas!", true);
echo greeting;
?>
```

59

PHP Constant Arrays

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

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

60

Constants are Global

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

```php
<?php
define("GREETING", "Welcome to TUPVisayas!");

function myTest() {
  echo GREETING;
}

myTest();
?>
```

61

PHP Operators

- Operators are used to perform operations on variables and values.
- PHP divides the operators in the following groups:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Array operators
  - Conditional assignment operators

62

PHP Arithmetic Operators

- The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

63

PHP Assignment Operators

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

64

PHP Comparison Operators

- The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

65

PHP Increment / Decrement Operators

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

66

PHP Logical Operators

- The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

67

PHP String Operators

- PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

68

PHP Array Operators

- The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

69

PHP Conditional Assignment Operators

- The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7

70

PHP if...else...elseif Statements

- Conditional statements are used to perform different actions based on different conditions.
- 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
  - switch statement - selects one of many blocks of code to be executed

71

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

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

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

72

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 - The switch Statement

- The switch statement is used to perform different actions based on different conditions.
- Use the switch statement to **select one of many blocks of code to be executed.**

77

PHP - The switch Statement

- Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

78

Example

```
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

79

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

80

PHP while Loop

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

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

81

Examples

- The example below displays the numbers from 1 to 5:

```
<?php
$x = 1;

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

82

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.
- Syntax

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

83

Example

```
<?php
$x = 1;

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

84

PHP for Loop

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

```
for (init counter; test counter; increment/decrement 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/decrement counter: Increases/decreases the loop counter value

Examples

- The example below displays the numbers from 0 to 10:

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


PHP foreach Loop

- The foreach loop - Loops through a block of code for each element 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):

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

Example

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

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

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

89

PHP Break

- 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:

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

90

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:

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

91