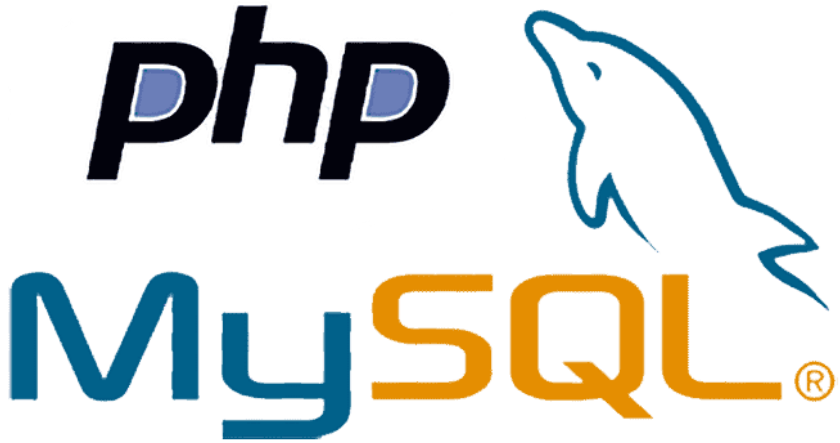
2021 – 2022



Introduction to PHP

Eng. Hazem A. Alrakhawi, Dr. Tawfiq S Barhoom

PHP Installation

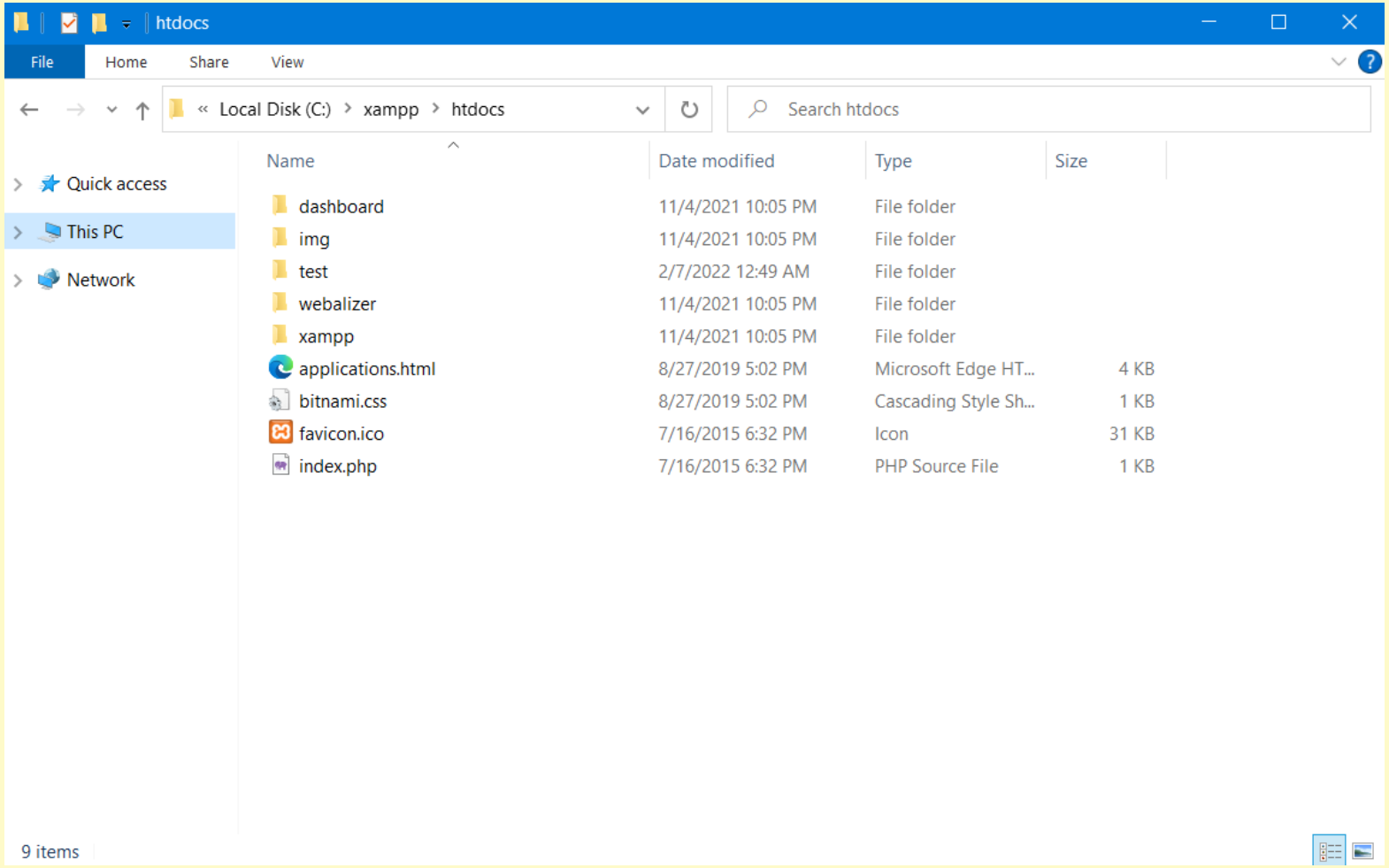
To start using PHP, you can:

- Find a web host with PHP and MySQL support as (**godaddy.com**, **bluehost.com**, **HostGator .com**).
- Install a web server on your own PC, and then install PHP and MySQL as (**XAMPP Server**).
- Install the IDE - **Visual Studio Code**.

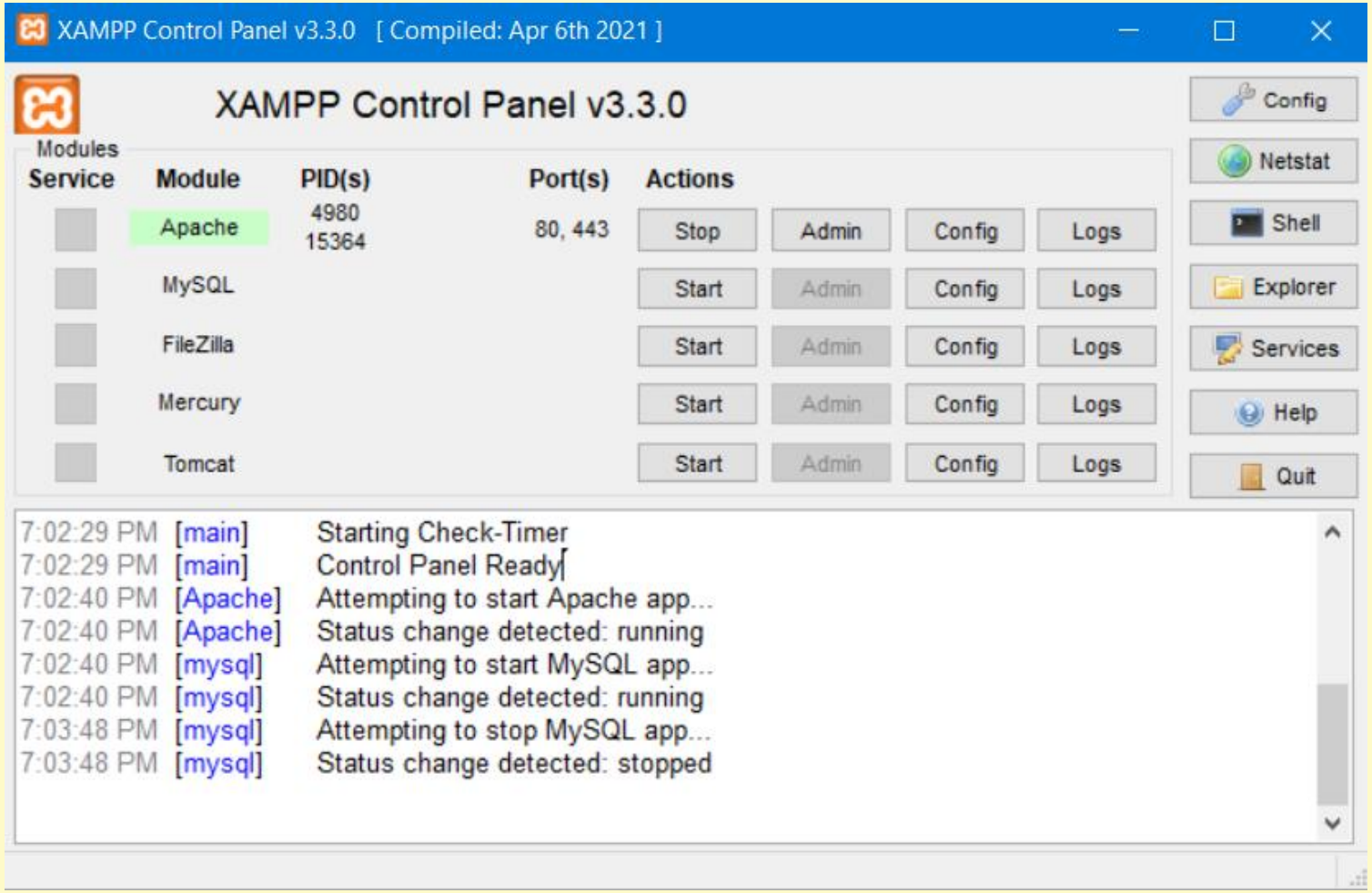
PHP Installation

- Just create some **.php files**, place them in your **web directory (htdocs) in XAMPP Server**, and the server will automatically parse them for you.
- You do not need to compile anything or install any extra tools.

PHP Installation



Run XAMPP and Start Apache Server



PHP SYNTAX

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
?>
```

Basic PHP Syntax

- The default file extension for PHP files is "**.php**".
- A PHP file normally contains **HTML tags**, and some **PHP scripting code**.
- PHP statements end with a **semicolon (;)**.

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

PHP Case Sensitivity

- 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>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

PHP Comments

- A comment in PHP code is a line that is not executed as a part of the program.
- Let others understand your code.
- Remind yourself of what you did.

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

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

# This is also a single-line comment
?>

</body>
</html>
```

PHP Comments

Example:

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

<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>

</body>
</html>
```

PHP Comments

Example:

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

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

</body>
</html>
```

PHP VARIABLES

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.

PHP Variables

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

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Rules for PHP variables:

- 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 _).

Output Variables

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

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

<?php
$txt = "Hazem Alrakhawi";
echo "I am $txt!";
?>

</body>
</html>
```

Output Variables

- The following example will produce the same output as the previous example.

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

<?php
$txt = "Hazem Alrakhawi";
echo "I am " . $txt . "!";
?>

</body>
</html>
```

Output Variables

- The following example will output the sum of two variables.

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

<?php
$x = 13;
$y = 45;
echo $x + $y;
?>

</body>
</html>
```

PHP VARIABLES SCOPE

PHP Variables Scope

PHP has three different variable scopes:

- local
- global
- static

Local Scope

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

Example:

```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>";
?>
```

Global Scope

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

Example:

```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>";
?>
```

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).

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

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

myTest();
echo $y; // outputs 15
?>
```

PHP The $GLOBALS Array

- **$GLOBALS** is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).
- PHP stores all global variables in an array called **$GLOBALS[index].**
- The **index holds the name of the variable**.

PHP The $GLOBALS Array

- 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.

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

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

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

myTest();
echo $y;
?>

</body>
</html>
```

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.
- use the **static keyword** *when you first declare the variable.*

```php
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
myTest();
myTest();
?>
```

PHP ECHO AND PRINT STATEMENTS

PHP echo and print Statements

- 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 arguments, while print can take one.
- echo is marginally **faster than** print.
- The echo statement can be used with or without parentheses: **echo or echo().**
- The print statement can be used with or without parentheses: **print or print().**

PHP echo Statements

Example:

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

<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "Eng. ", "Hazem ", "A. ", "Alrakhawi ";
?>

</body>
</html>
```

PHP echo Statements

Example:

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

<?php
$txt1 = "PHP Course";
$txt2 = "Eng. Hazem Alrakhawi";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Trainer name is " . $txt2 . "<br>";
echo $x + $y;
?>

</body>
</html>
```

PHP print Statements

Example:

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

<?php
$txt1 = "PHP Course";
$txt2 = "Eng. Hazem Alrakhawi";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Trainer name is " . $txt2 . "<br>";
print $x + $y;
?>

</body>
</html>
```

PHP DATA TYPES

PHP Data Types

- **PHP supports the following data types:**
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP Data Types

- The **var_dump()** function is used to dump information about a variable.
- The PHP **var_dump()** function returns the **data type** and **value**.

```php
<?php
$x = 43;
var_dump($x);

$y = true;
var_dump($y);

$z = 3.9;
var_dump($z);

$c = "Hazem ";
var_dump($c);

?>
```

PHP CONSTANTS

PHP Constants

- 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**.

Create a PHP Constant

- **Example** - Create a constant with a **case-sensitive name**:

```php
<?php
// case-sensitive constant name
define("GREETING", "Welcome Hazem Alrakhawi");
echo GREETING;
?>
```

- **Example** - Create a constant with a **case-insensitive name**:

```php
<?php
// case-insensitive constant name
define("GREETING", "Welcome Hazem Alrakhawi", true);
echo greeting;
?>
```

Constants are Global

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

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

<?php
define("GREETING", "Welcome Hazem Alrakhawi");

function myTest() {
  echo GREETING;
}

myTest();
?>

</body>
</html>
```

PHP Constant Arrays

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

```php
<?php
define("cars", [
  "Ford",
  "Hyundai",
  "Toyota"
]);
echo cars[0];
?>
```

PHP OPERATORS

PHP Operators

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.

PHP Arithmetic Operators

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

PHP Arithmetic Operators

Example:

```php
<?php
$x = 2;
$y = 3;

echo $x ** $y;
?>
```

PHP Assignment Operators

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

PHP Assignment Operators

Example:

```php
<?php
$x = 15;
$x %= 4;

echo $x;
?>
```

PHP Comparison Operators

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.

PHP Comparison Operators

Example:

```php
<?php
$x = 100;
$y = "100";

echo($x === $y); //blank
echo "<br>";
var_dump($x === $y); // returns bool(false) because types are not equal
echo "<br>";
var_export($x === $y); // returns false because types are not equal

?>
```

PHP Comparison Operators

Example:

```php
<?php
$x = 100;
$y = "100";

var_dump($x <> $y); // returns bool(false) because values are equal
echo "<br>";
var_export($x <> $y); // returns false because values are equal
?>
```

PHP Comparison Operators

Example:

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

echo ($x <=> $y); // returns -1 because $x is less than $y
echo "<br>";

$x = 10;
$y = 10;

echo ($x <=> $y); // returns 0 because values are equal
echo "<br>";

$x = 15;
$y = 10;

echo ($x <=> $y); // returns +1 because $x is greater than $y
?>
```

PHP Increment / Decrement Operators

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

PHP Logical Operators

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

PHP String Operators

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

PHP Array Operators

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

PHP Conditional Assignment Operators

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

PHP Conditional Assignment Operators

Example:

```php
<?php
    // if empty($user) = TRUE, set $status = "anonymous"
    echo $status = (empty($user)) ? "anonymous" : "logged in";
    echo("<br>");

    $user = "John Doe";
    // if empty($user) = FALSE, set $status = "logged in"
    echo $status = (empty($user)) ? "anonymous" : "logged in";
?>
```

PHP Conditional Assignment Operators

Example:

```php
<?php
   // variable $user is the value of $_GET['user']
   // and 'anonymous' if it does not exist
   echo $user = $_GET["user"] ?? "anonymous";
   echo("<br>");

   // variable $color is "red" if $color does not exist or is null
   echo $color = $color ?? "red";
?>
```

PHP Conditional Assignment Operators

Example:

```php
<?php
    // variable $user is the value of $_GET['user']
    // and 'anonymous' if it does not exist
    echo $user = $_GET["user"] ?? "anonymous";
    echo("<br>");

    // variable $color is "red" if $color does not exist or is null
    $color = "blue";
    echo $color = $color ?? "red";
?>
```

PHP Conditional Assignment Operators

Example:

```php
<?php
   // variable $user is the value of $_GET['user']
   // and 'anonymous' if it does not exist
   echo $user = $_GET["user"] ?? "anonymous";
   echo("<br>");

   // variable $color is "red" if $color does not exist or is null
   $color = null;
   echo $color = $color ?? "red";
?>
```

PHP STRING FUNCTIONS

PHP String Functions

- The PHP **strlen("Hazem")** function returns the length of a string.
- The PHP **str_word_count("Hazem Alrakhawi")** function counts the number of words in a string. **// outputs 2**
- The PHP **strrev("Hazem")** reverses a string. **// outputs mezaH**
- 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.
- **echo strpos("Hello world!", "world"); // outputs 6**

PHP String Functions

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

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

<?php
echo str_replace("world", "Hazem", "Hello world!");
//output is Hello Hazem!
?>

</body>
</html>
```

PHP MATH FUNCTIONS

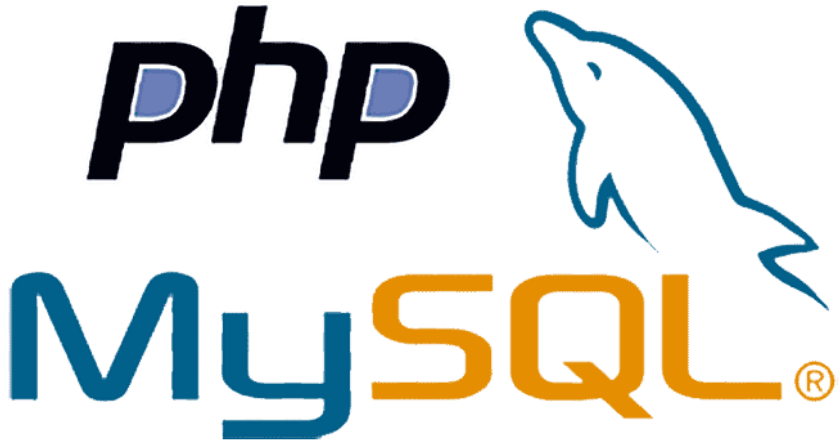
PHP Math Functions

- echo(**pi()**); // returns 3.1415926535898
- echo(**min(0, 17, 33, 10, -6, -200)**); // returns -200
- echo(**max(0, 170, 33, 20, -6, -200)**); // returns 170
- echo(**abs(-6.7)**); // returns 6.7
- echo(**round(1.65)**); // returns 2
- echo(**round(1.49)**); // returns 1
- echo(**rand()**); //returns random number as 2015248788

PHP Math Functions

- echo(**rand(1, 100)**); //returns random integer between 1 and 100 (**inclusive**).
- echo(**floor(4.6)**); Round numbers down to the nearest integer.
- echo(**ceil(5.1)**); Round numbers up to the nearest integer.
- echo(**pow(2,3)**); Returns x raised to the power of y.
- echo(**sqrt(9)**); Return the square root of the number x.

2021 - 2022



The End.

Eng. Hazem A. Alrakhawi, Dr.Tawfiq S Barhoom