



Faculty of Science and Technology  
Department of Computer Science

# WEB TECHNOLOGIES (CSC 3222)

Lecture Note 2

Week 2

Supta Richard Philip  
Richard@aiub.edu

# Contents

Introduction to PHP .....	3
What is PHP .....	3
PHP Installation.....	4
First PHP Script.....	5
PHP Syntax and variable .....	6
Comments .....	7
Variables .....	7
String .....	8
Integer .....	8
Float .....	8
Boolean .....	8
Array .....	9
Object.....	9
Null .....	9
Built-in Array Function .....	9
Conditional statement.....	11
If else .....	11
Switch .....	11
Iteration (Loop) .....	12
While loop .....	12
Do while loop .....	12
For loop.....	13
Foreach loop .....	13
Function.....	14
Class and Object.....	16
References .....	18



# Introduction to PHP

In this note, we will discuss more details about "PHP: Hypertext Preprocessor"; advantage and importance of PHP. We will also discuss about PHP as language and its basic data structure.

The key point we will learn in this note as follows:

- What is PHP
- Importance of PHP for developing web application as business context
- Install PHP in your development environment
- First PHP script HelloWorld.php
- Different data types and variables in PHP; Array and Object
- Conditional statement in PHP
- Iteration using PHP (Loop)
- Function
- Class and Object

## What is PHP

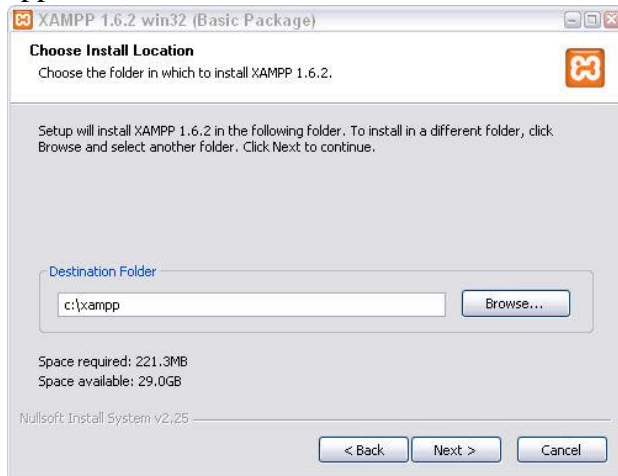
- PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
- PHP 7 is the latest stable release.
- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP scripts are executed on the server
- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"
- 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

# PHP Installation

XAMPP is an all-in-one package that you can use to install Apache, PHP, and MySQL. XAMPP also installs phpMyAdmin, a Web application used to administer MySQL. XAMPP installs all the packages in one easy procedure.

Installing XAMPP on Windows step by step

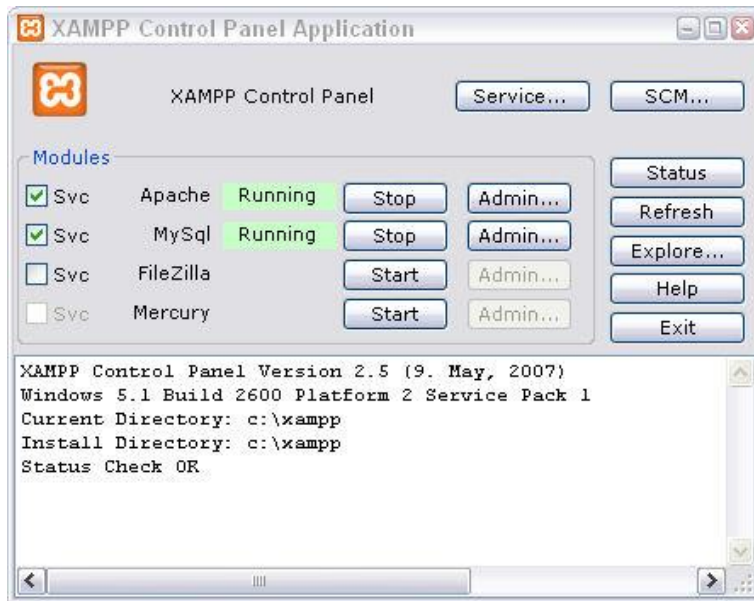
1. Go to [www.apachefriends.org/en/xampp-windows.html](http://www.apachefriends.org/en/xampp-windows.html).
2. Download installer and Double-click the file name. The Installation Wizard starts.
3. Read and click through the next few windows until the Choose Install Location screen appears, as shown below.



4. Under the "SERVICE SECTION", check Apache and MySQL, as shown below. This installs the tools as Windows services.



- Click Install. The installation process takes a few minutes to complete and Click Finish.
- The open Control Panel with Apache and MySQL running is shown and start the Apache and mysql service as below.



- Your Server is up and running.

## First PHP Script

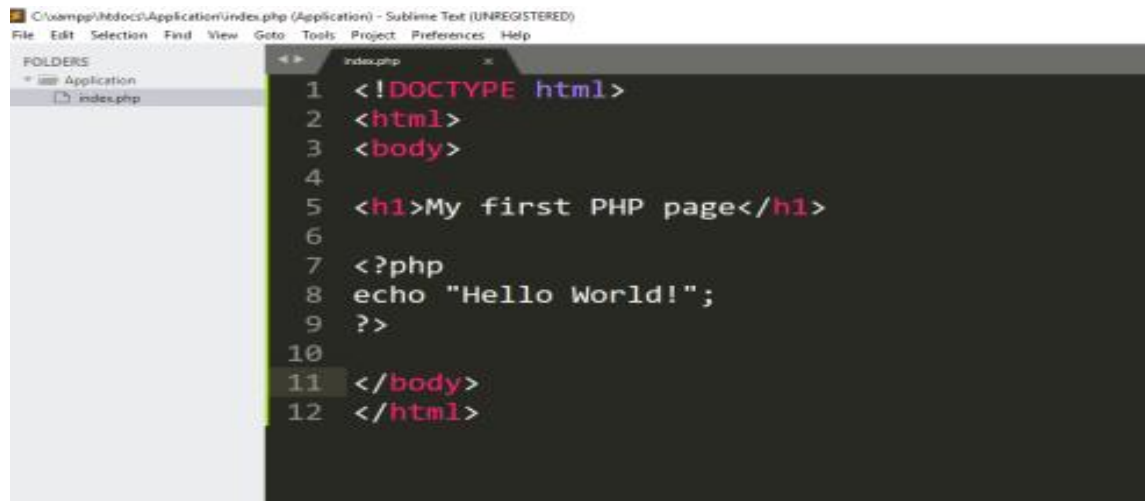
The very first program of any programming language is **Hello World**. In this section, we will also write our first PHP script and run on the server.

Basic PHP Hello World as follows:

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

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

Use any text editor to write the code. The default file extension for PHP files is **".php"**. A PHP file normally contains HTML tags, and some PHP scripting code.



```

C:\xampp\htdocs\Application\index.php (Application) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
+ Application
  index.php

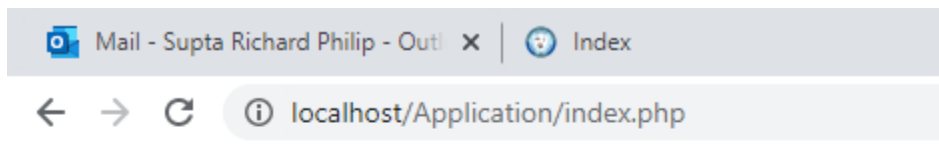
index.php
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>My first PHP page</h1>
6
7 <?php
8 echo "Hello World!";
9 ?>
10
11 </body>
12 </html>

```

Go to your XAMP installation directory, e.g. C:\xampp\htdocs and create a new directory on it.

For example your directory name is **Application**. Now save the file as **HelloWorld.php** in **C:\xampp\htdocs\Application** location.

Start the server and browse **http://localhost/Application/HelloWorld.php** in the browser.



## My first PHP page

Hello World!

## PHP Syntax and variable

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:

```

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

```

But 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:

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

## Comments

PHP single-line comment.

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

# This is also a single-line comment
?>
```

PHP multiple-line comments.

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

## Variables

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

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

### 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)



**Variables are as containers for storing data. PHP variable names are case-sensitive. We do not declare data type before the variables. PHP is a Loosely Typed Language (not strictly type: C, Java, C#) means PHP automatically associates a data type to the variables.**

More or less PHP supports the following data types and we will see the example one by one.

- String
- Integer
- Float (also called double)
- Boolean
- Array
- Object
- Null

## String

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

## Integer

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

## Float

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

## Boolean

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

## Array

It is convenient to use `var_dump()` to display the detail content of an array, which is handy in debugging.

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

## Object

We will discuss more about class and object on later.

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}
// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

## Null

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

## Built-in Array Function

### Built-in Array Functions

- **`is_array($var)`**: returns TRUE if `$var` is an array.
- **`count($array)`**: returns the number of elements in the given array.
- **`in_array($value, $array)`**: returns TRUE if the array contains the given value.
- **`sort($array)`, `sort($array, SORT_NUMERIC)`, `sort($array, SORT_STRING)`**: sort the elements of the given array. `sort()` operates on the array directly (by reference).
- **`rsort($array, SORT_NUMERIC)`, `rsort($array, SORT_STRING)`**: sort in *reverse* order.

- **shuffle(\$array)**: put the elements in random order. `shuffle()` also operates on the array directly.
- **implode(\$glue, \$array)**: join the array elements with the `$glue` string. For example,

```
$a = array('apple', 'orange', 'banana');
echo implode('|', $a);

// apple|orange|banana
```

- **explode(\$delimiter, \$str)**: takes a string and tokenizes into an array based on the delimiter.

```
<?php
$days='Mon,Tus,Wed,Thu';
$dayArray = explode(',', $days);
var_dump($dayArray);
?>

array (size=4)
  0 => string 'Mon' (length=3)
  1 => string 'Tus' (length=3)
  2 => string 'Wed' (length=3)
  3 => string 'Thu' (length=3)
```

- **array\_push(\$array, \$value1, \$value2,...)**: pushes one or more elements to the end of the array. If there is only one element, you can use "`$array[] = value`".
- **array\_pop(\$array)**: pops the last element off the array.
- **array\_shift(\$array)**: shift an element off the beginning of the array.
- **array\_unshift(\$array, \$value1, \$value2,...)**: prepend one or more elements at the beginning of the array.
- **reset(\$array), end(\$array)**: returns the first element and last element of the array, respectively.

# Conditional statement

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

## If else

### Example 1

```
<?php
$age = 15
if ($age < 18) {
    echo "You are not adult";
}
?>
```

### Example 2

```
<?php
$age = 15
if ($age < 18) {
    echo "You are not adult";
} else {
    echo "You are adult";
}
?>
```

### Example 3

```
<?php
$age = 15
if ($age < 18) {
    echo "You are not adult";
} elseif($age < 40) {
    echo "You are adult";
} else {
    echo "You are old";
}

?>
```

## Switch

```
<?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!";
}
?>

```

## Iteration (Loop)

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

## While loop

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

```

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

```

## 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.

```

<?php
$x = 1;

do {

```

```

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

```

## For loop

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

```

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

```

## Foreach loop

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

```

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

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

```

## Exercise:

1. **Write a PHP script to calculate the area and perimeter of a Rectangle, and display the result.**

Hints: The area of a Rectangle = length  $\times$  width, perimeter =  $2 \times (\text{length} + \text{width})$

2. **Write a PHP script to calculate the VAT (Value Added Tax) over an amount**

Hints: VAT = 15% of the amount

3. **Write a PHP script to find whether a given number is odd or even.**

Hints: use IF-ELSE

4. **Write a PHP script to find the largest number from three given numbers.**

Hints: use IF-ELSE

5. **Write a PHP script to print all the odd numbers between 10 to 100**

Hints: use LOOP & IF-ELSE

## Use the following built in function when you solve the exercise.

To find the type of a variable, use function **gettype()** built-in function.

You can also use boolean function **is\_xxx(var)** (e.g., **is\_integer()**, **is\_int()**, **is\_double()**, **is\_string()**, **is\_array()**, **is\_object()**) to check if a variable belongs to a certain type.

For example, `<?php echo gettype($var), '<br>'; ?>`

`<?php echo is_integer($var), '<br>'; ?>`

## Checking Variables

You can use the following built-in functions:

**isset(\$var)**: return TRUE if \$var is set and is not NULL.

**unset(\$var)**: unset the \$var.

**is\_null(\$var)**: return TRUE if the \$var is NULL.

**empty(\$var)**: return TRUE if the \$var does not exist (i.e. **!isset(\$var)**) or its value is equivalent to boolean FALSE (e.g., NULL, 0, 0.0, "", '0', empty array and object).

## Function

PHP has more than **1000 built-in functions**, and in addition you can create your own custom functions.

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, 5);
?>
```

### Example: Pass by Reference

```
// $v2 passed by reference
function funByRef($v1, &$v2) {
    $v1 = $v1 * 2;
    $v2 = $v2 * 2;
    var_dump($v1); // int(6)
    var_dump($v2); // int(10)
}
$a = 3;
$b = 5;
funByRef($a, $b);
var_dump($a); // int(3) - no change
var_dump($b); // int(10) - change thru the function
```

PHP is a Loosely Typed Language. In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding the strict declaration, it will throw a "Fatal Error" if the data type mismatches.

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

To specify `strict` we need to set `declare(strict_types=1);`.

```
<?php declare(strict_types=1); // strict requirement

function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will be thrown
?>
```

## Default Argument Value

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

## Returning values

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
```



```

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>

```

## Return Type Declarations

```

<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>

```

# Class and Object

From PHP5, you can also write PHP code in an object-oriented style. Object-Oriented programming is faster and easier to execute. Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

A class is a template for objects, and an object is an instance of class.

*// Fruit.php*

```

<?php
class Fruit {
    // Properties
    private $name;
    private $color;

    //constructor
    public function __construct($name,$color) {
        $this->name = $name;
        $this->color = $color;
    }
    //destructor
    public function __destruct() {
        echo "destructor";
    }
}

```

```

    }

    // Methods
    public function set_name($name) {
        $this->name = $name;
    }
    public function get_name() {
        return $this->name;
    }

    public function set_color($color) {
        $this->color = $color;
    }
    public function get_color() {
        return $this->color;
    }

    public function __toString() {
        return __CLASS__ . '[name = ' . $this->name . 'color=' . $this->color . ']';
    }
}

?>

```

### *// FruitTest.php*

```

<?php
$apple = new Fruit("Apple","Green");
echo $apple->get_name();
echo $apple->get_color();
echo "$apple\n";

?>

```

## Exercise

1. Write a PHP class using the following UML and write a test case.

MyCircle
- \$radius: number
+ __construct(\$radius: number=1): void
+ __destruct(): void
+ getRadius(): number
+ setRadius(\$radius: number): void
+ getArea(): number
+ __toString(): string

## References

- [1]. [https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/php5\\_OOP.html](https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/php5_OOP.html)
- [2]. [https://www.w3schools.com/php/php\\_oop\\_what\\_is.asp](https://www.w3schools.com/php/php_oop_what_is.asp)