



# HiLCoE School of Computer Science & Technology

## Chapter One: Basics Of PHP

**Course Title : Web Technologies II**

**Instructor name: Yitayew Solomon**

**E-mail address: [yitayewsolomon3@gmail.com](mailto:yitayewsolomon3@gmail.com)**



# SERVER-SIDE SCRIPTING



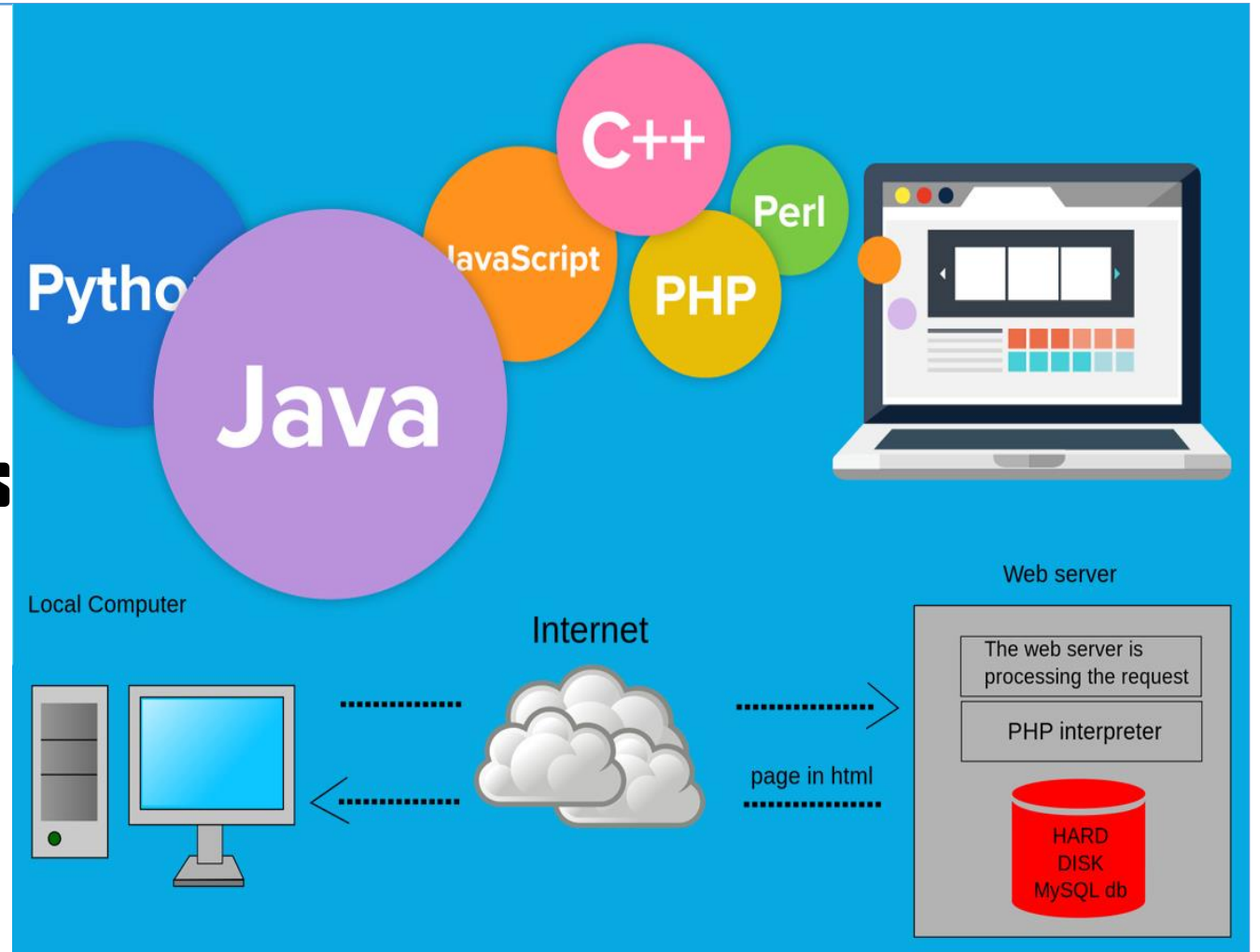
# Outlines

📌 Server Side Scripting

📌 How SSS Works

📌 Common SSS Languages

📌 PHP SETUP



# Server Side Scripting

- A server-side scripting language is a programming language used to create scripts that run on a **web server** to generate dynamic web content before the page is sent to the user's web browser.
- These scripts interact with the **server's resources**, such as **databases** and **file systems**, to perform tasks like retrieving, processing, and sending data.
- Server-side scripts are **executed** on the **server** rather than the client's browser, which allows for more complex operations and data processing.

# Key Features of Server-Side Scripting Languages

- ❖ **Dynamic Content Generation:** Server-side scripts can generate dynamic web pages that change based on user interactions or other variables. This allows for personalized and interactive user experiences.
- ❖ **Database Interaction:** These languages can communicate with databases to retrieve, store, and manipulate data. This is essential for creating web applications like **e-commerce sites, social networks, and content management systems.**

# Cont. ...

- ❖ **Security:** Server-side scripting can manage sensitive data and perform secure transactions. Since the code runs on the server, it can be protected from direct access by users.
- ❖ **Session Management:** Server-side scripts can handle user sessions and maintain state across multiple pages or visits, enabling features like user login systems and shopping carts.

## Cont. ...

❖ **Business Logic:** These scripts handle the core business logic of web applications, such as processing form submissions, validating input, and performing calculations.

# How Server-Side Scripting Works

## ❖ Client Request:

- A user requests a web page by entering a URL or clicking a link. This request is sent to the web server.

## ❖ Server Processing:

- The web server receives the request and passes it to the appropriate server-side script. The script processes the request, which may involve interacting with a database, performing calculations, or fetching data.



# Cont. ...

## ❖ **Dynamic Content Generation:**

- The server-side script generates the HTML (or other content types) based on the processed data and the request parameters.

## ❖ **Response Sent to Client:**

- The web server sends the generated HTML back to the client's web browser, which renders the page for the user to view.

# Top Server Side Scripting Language

## ❖ PHP

### • Features:

- ✓ Widely used for web development.
- ✓ Embedded directly into HTML.
- ✓ Extensive library support.
- ✓ Cross-platform compatibility.
- ✓ Strong community support
- ✓ Extensive documentation.
- ✓ Integrates well with various databases.

## Reasons Why PHP Is An Excellent Option For Web Development



# Cont. ...

- **Limitations:**

- ✓ Security vulnerabilities due to improper coding.
- ✓ Can be slower than other languages due to its interpreted nature.
- ✓ Lack of strong typing can lead to runtime errors.

# Node.js (JavaScript)

- **Features:**

- ✓ Event-driven, non-blocking I/O model.
- ✓ High performance for real-time applications.
- ✓ Uses JavaScript, making it easier for front-end developers to transition.
- ✓ Rich ecosystem with npm.
- ✓ Scalable for microservices and single-page applications.



[www.metricsviews.com](http://www.metricsviews.com)

## Cont. ...

- **Limitations:**

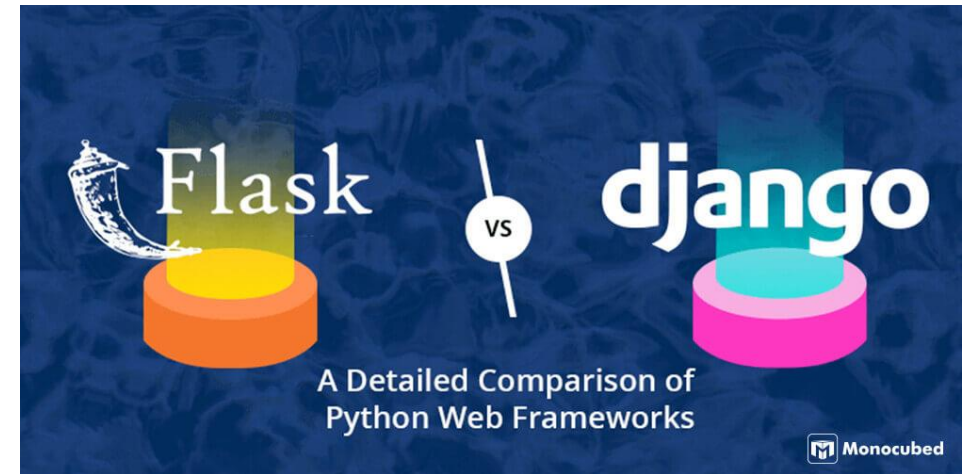
- ✓ Callback hell due to asynchronous nature.
- ✓ Single-threaded, which can be a bottleneck for CPU-intensive operations.
- ✓ Requires a good understanding of asynchronous programming.



# Python (Django, Flask)

- **Features:**

- ✓ High readability and simple syntax.
- ✓ Extensive libraries and frameworks.
- ✓ Strong support for data science and AI applications.
- ✓ Highly scalable and versatile.
- ✓ Django and Flask provide rapid development capabilities.



# Cont. ...

- **Limitations:**

- ✓ Slower performance compared to compiled languages.
- ✓ Can be less suitable for mobile computing.
- ✓ Global Interpreter Lock (GIL) can be a limitation for multi-threaded applications.

# Java (Spring)

- **Features:**

- ✓ Strongly typed and object-oriented.
- ✓ High performance with just-in-time compilation.
- ✓ Platform-independent due to JVM.
- ✓ Robust, secure, and scalable.
- ✓ Large ecosystem with extensive libraries and frameworks.



# Cont. ...

- **Limitations:**

- ✓ Verbose syntax.
- ✓ Steeper learning curve.
- ✓ Requires more resources for development and deployment.

# C# (ASP.NET)

## Features:

- ✓ Modern, object-oriented language.
- ✓ Integrated with the .NET framework.
- ✓ High performance with compiled code.
- ✓ Strong support for Windows-based applications.
- ✓ Excellent tooling with Visual Studio.





## Cont. ...

- **Limitations:**

- ✓ Primarily suited for Windows environments.
- ✓ Licensing costs for enterprise usage.
- ✓ Less flexibility compared to open-source solutions.

# Cont. ...

## Summary Table

Language	Features	Limitations
<b>PHP</b>	Widely used, embedded in HTML, extensive libraries, cross-platform, strong community	Security vulnerabilities, slower performance, lack of strong typing
<b>Node.js</b>	Event-driven, high performance, JavaScript, rich ecosystem, scalable	Callback hell, single-threaded limitations, requires async programming
<b>Python</b>	Readable syntax, extensive libraries, strong data science support, scalable, versatile	Slower performance, less suitable for mobile, GIL limitations

# Cont. ...

<b>Ruby</b>	Convention over configuration, productive, strong community, easy to read/write, rapid dev	Slower runtime, memory consumption, challenging deployment at scale
<b>Java</b>	Strong typing, high performance, platform-independent, robust, large ecosystem	Verbose syntax, steep learning curve, resource-intensive
<b>C#</b>	Modern, object-oriented, integrated with .NET, high performance, excellent tooling	Primarily for Windows, licensing costs, less flexibility

# Cont. ...

<b>Perl</b>	Strong text processing, flexible, mature, CPAN modules, cross-platform	Complex syntax, performance issues, less popular for modern web dev
<b>ColdFusion</b>	Rapid development, protocol support, built-in functions, scalable, secure	Smaller community, licensing costs, less popular
<b>Go</b>	High performance, concurrency support, simple syntax, strong standard library, scalable	Limited libraries, verbose error handling, smaller community
<b>Rust</b>	Memory safety, high performance, concurrency support, modern syntax, growing ecosystem	Steep learning curve, slower compilation, less mature web dev libraries







# PHP

- PHP (Hypertext Preprocessor) is a widely-used **open-source server-side** scripting language designed for web development.
- It is particularly suited for creating **dynamic** and **interactive** websites.
- PHP code is embedded within HTML, and when a webpage with PHP code is requested, the server processes the PHP code and outputs HTML to the user's browser.

# key features of PHP

- ❖ **Server-Side Scripting:** PHP runs on the server, and the client (browser) only sees the output (HTML), not the PHP code.
- ❖ **Database Integration:** PHP can interact with databases like MySQL, PostgreSQL, and others, making it ideal for building database-driven websites.
- ❖ **Cross-Platform:** PHP works on various platforms (Windows, Linux, macOS).
- ❖ **Free and Open-Source:** PHP is free to use and modify, with a large community of developers contributing to its improvements.

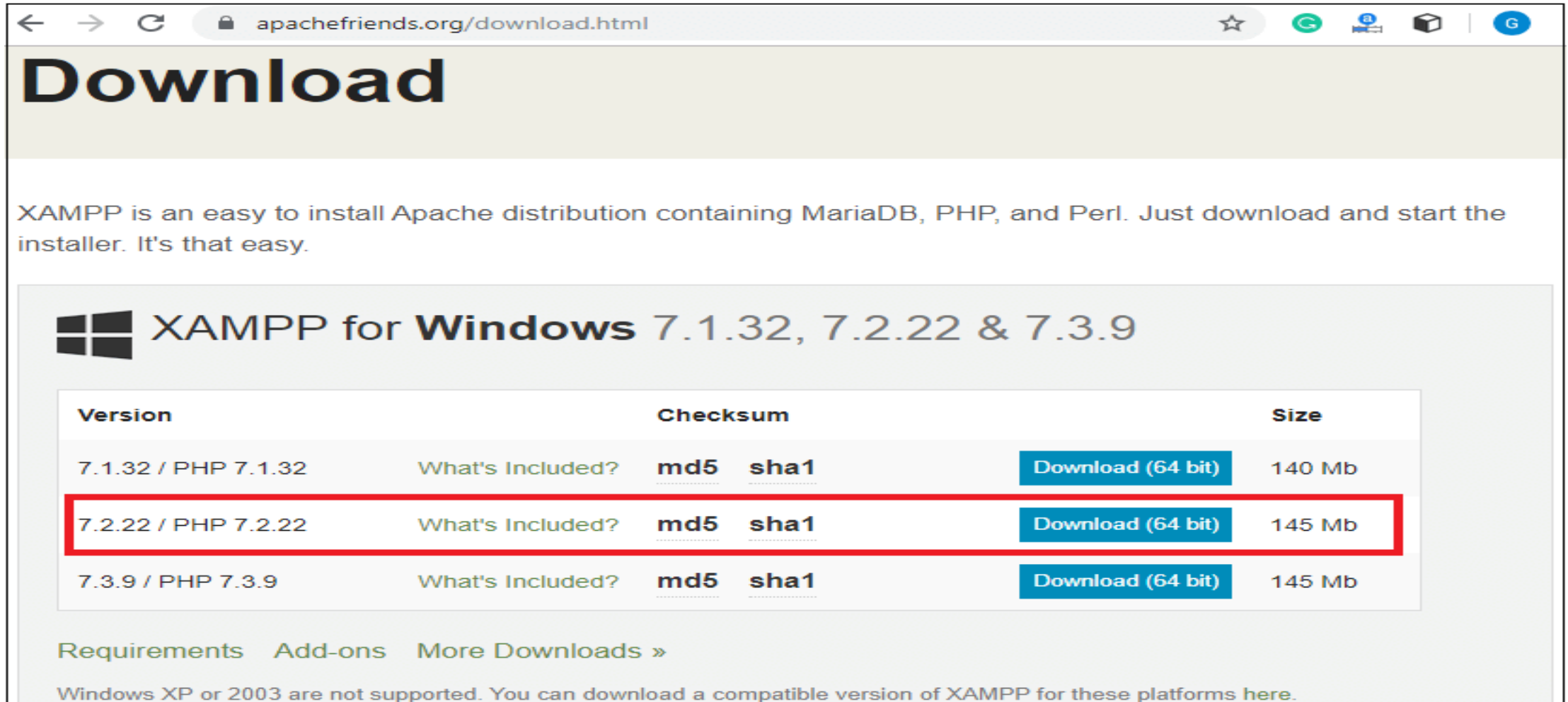
# PHP Installation

## Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- **WAMP** for Windows
- **LAMP** for Linux
- **MAMP** for Mac
- **SAMP** for Solaris
- **FAMP** for FreeBSD
- **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, Mercury Mail, etc.

# Download



The screenshot shows the Apache Friends download page. At the top, the browser address bar displays 'apachefriends.org/download.html'. Below the browser, the page has a large 'Download' header. A paragraph states: 'XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.' Below this, a section titled 'XAMPP for Windows 7.1.32, 7.2.22 & 7.3.9' features a table of download links. The table has columns for Version, Checksum (md5 and sha1), and Size. The row for version 7.2.22 / PHP 7.2.22 is highlighted with a red border. Below the table are links for 'Requirements', 'Add-ons', and 'More Downloads »'. A footer note states: 'Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here.'

Version	Checksum	Size
7.1.32 / PHP 7.1.32	What's Included? <a href="#">md5</a> <a href="#">sha1</a>	<a href="#">Download (64 bit)</a> 140 Mb
7.2.22 / PHP 7.2.22	What's Included? <a href="#">md5</a> <a href="#">sha1</a>	<a href="#">Download (64 bit)</a> 145 Mb
7.3.9 / PHP 7.3.9	What's Included? <a href="#">md5</a> <a href="#">sha1</a>	<a href="#">Download (64 bit)</a> 145 Mb

[Requirements](#) [Add-ons](#) [More Downloads »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

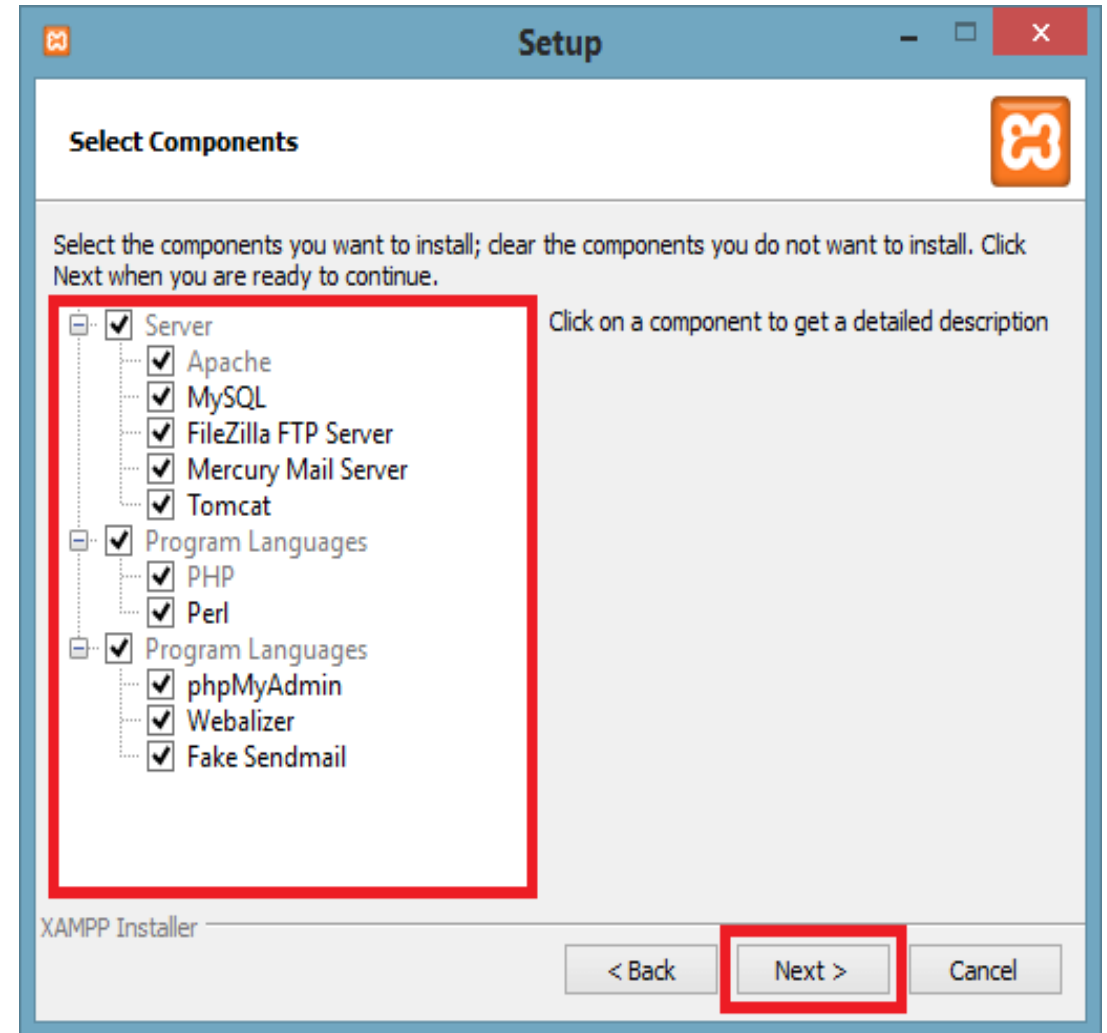
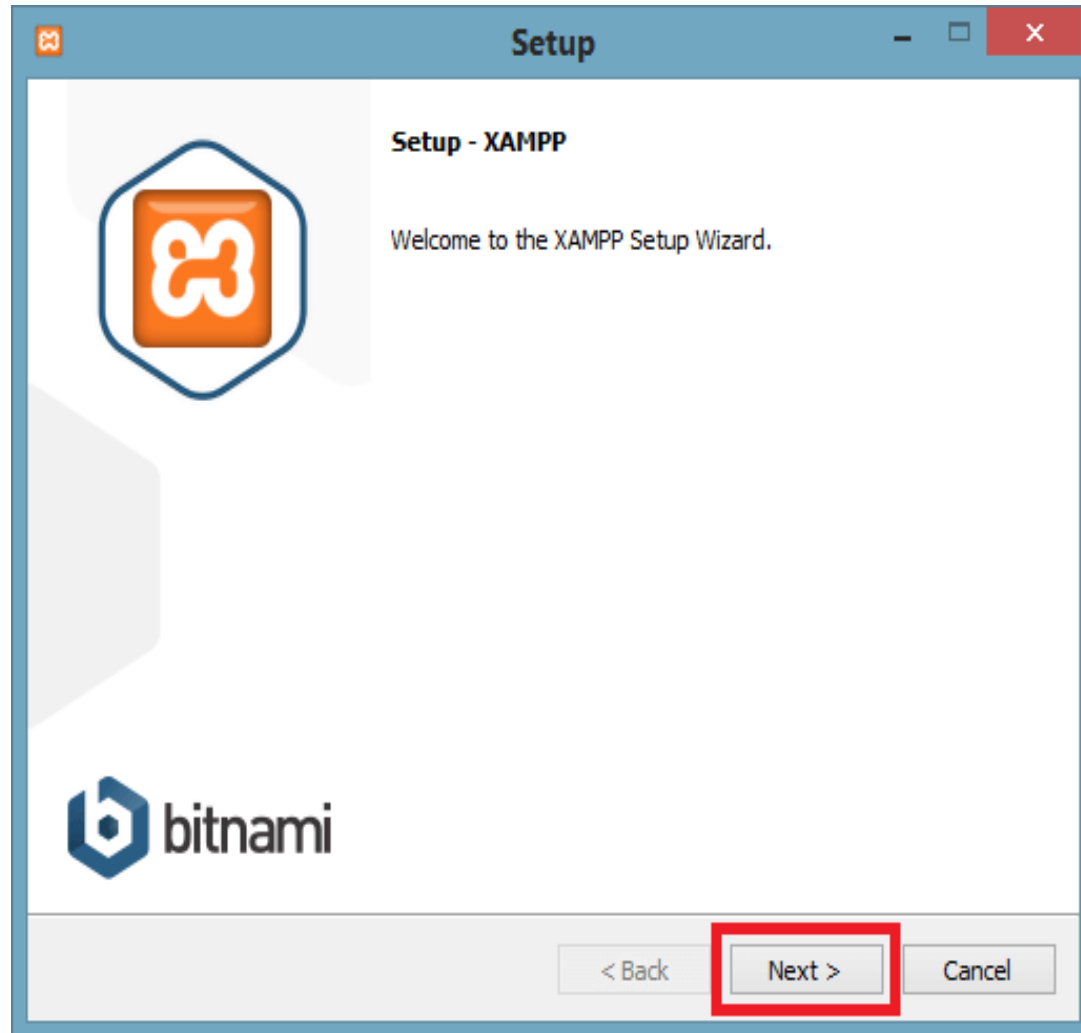
Source: <https://www.apachefriends.org/download.html>

## Cont. ...

- **Step 2:** After downloading XAMPP, double click on the downloaded file and allow XAMPP to make changes in your system. A window will pop-up, where you have to click on the Next button.
- **Step 3:** Here, select the components, which you want to install and click Next.



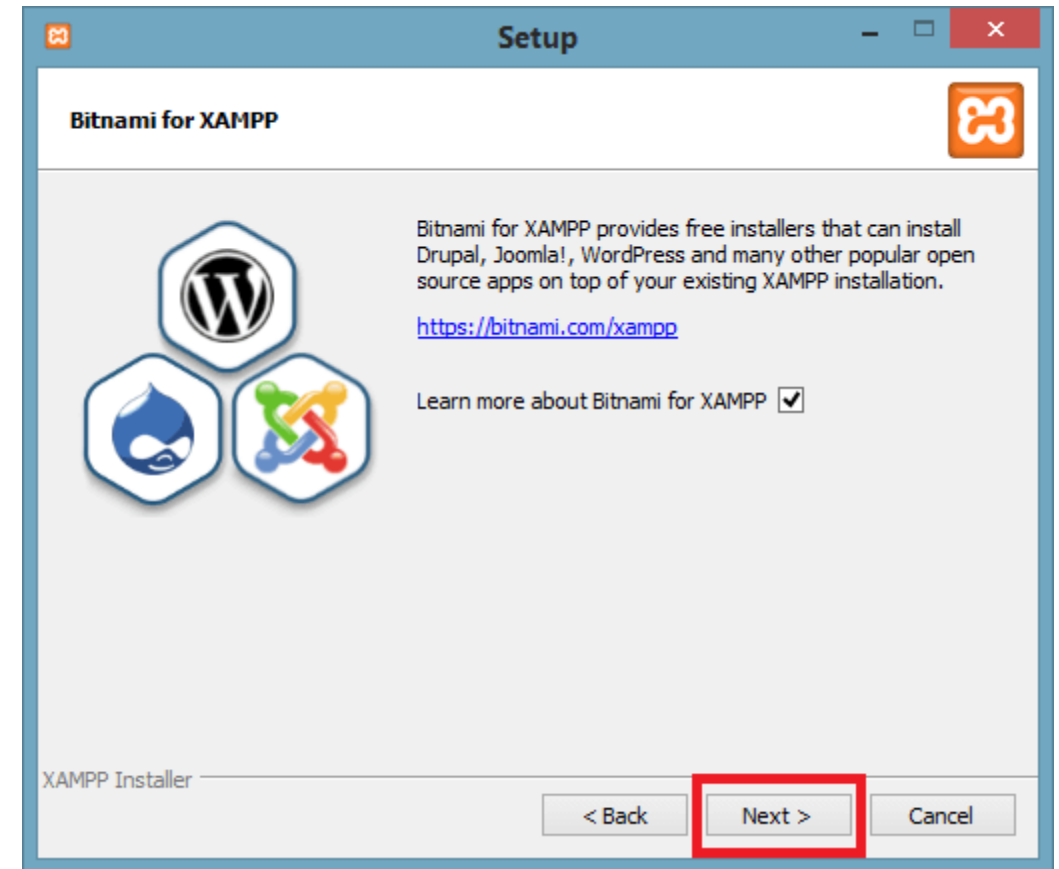
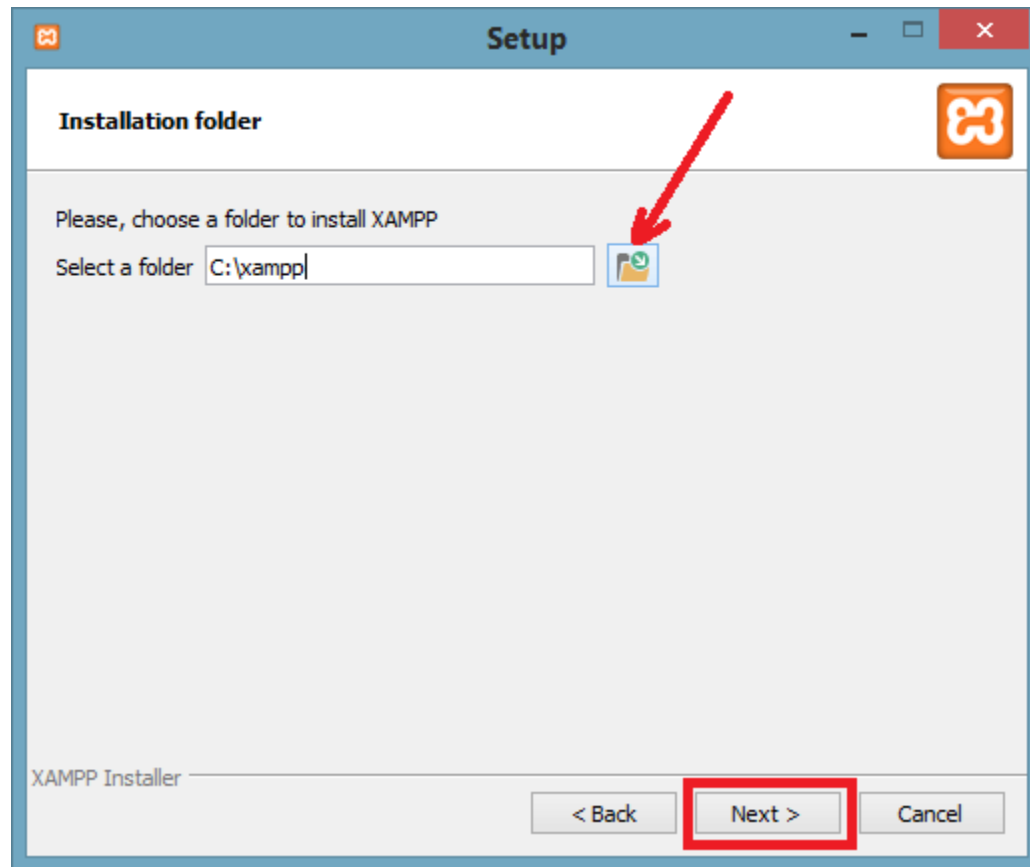
# Cont. ...



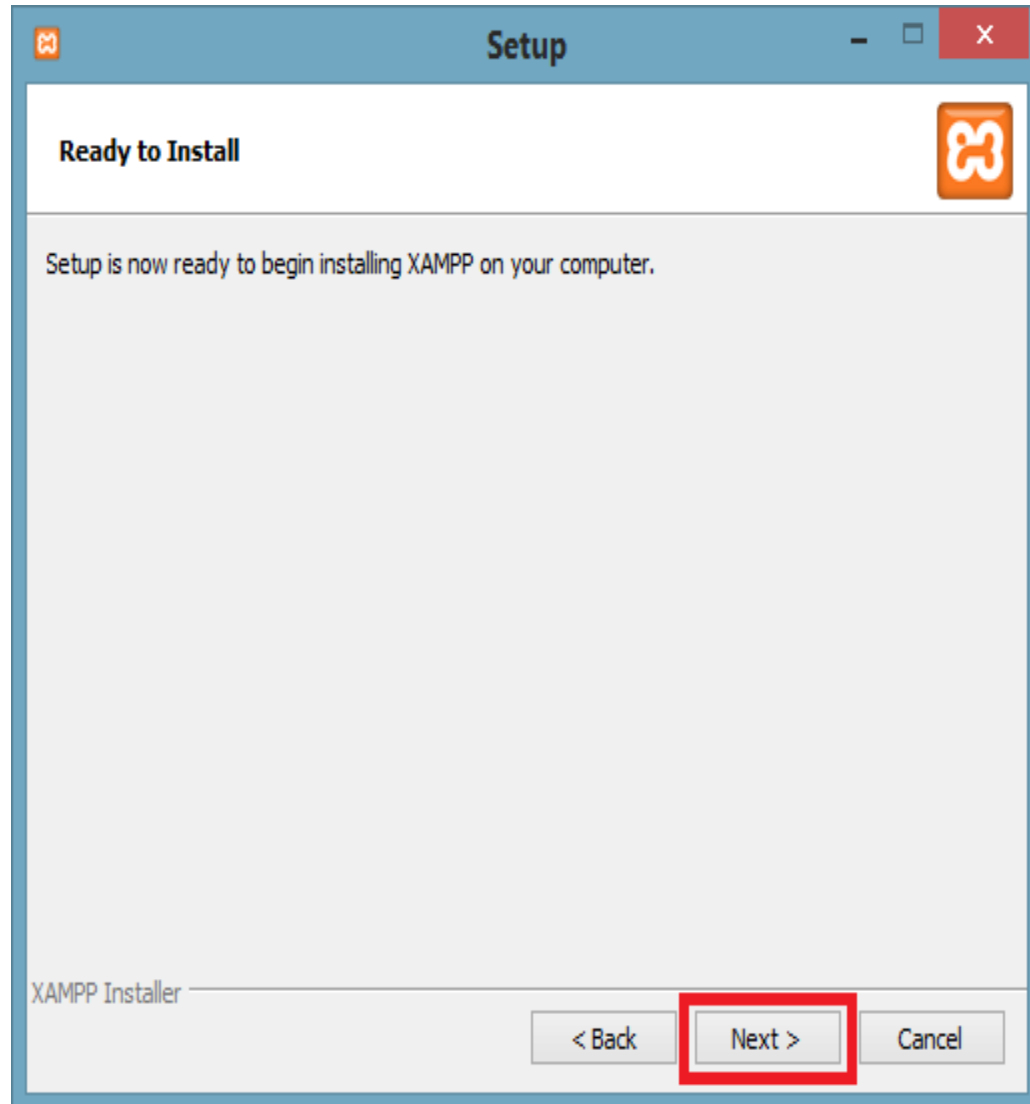
## Cont. ...

- **Step 4:** Choose a folder where you want to install the XAMPP in your system and click Next.
- **Step 5:** Click Next and move ahead.
- **Step 6:** XAMPP is ready to install, so click on the Next button and install the XAMPP.

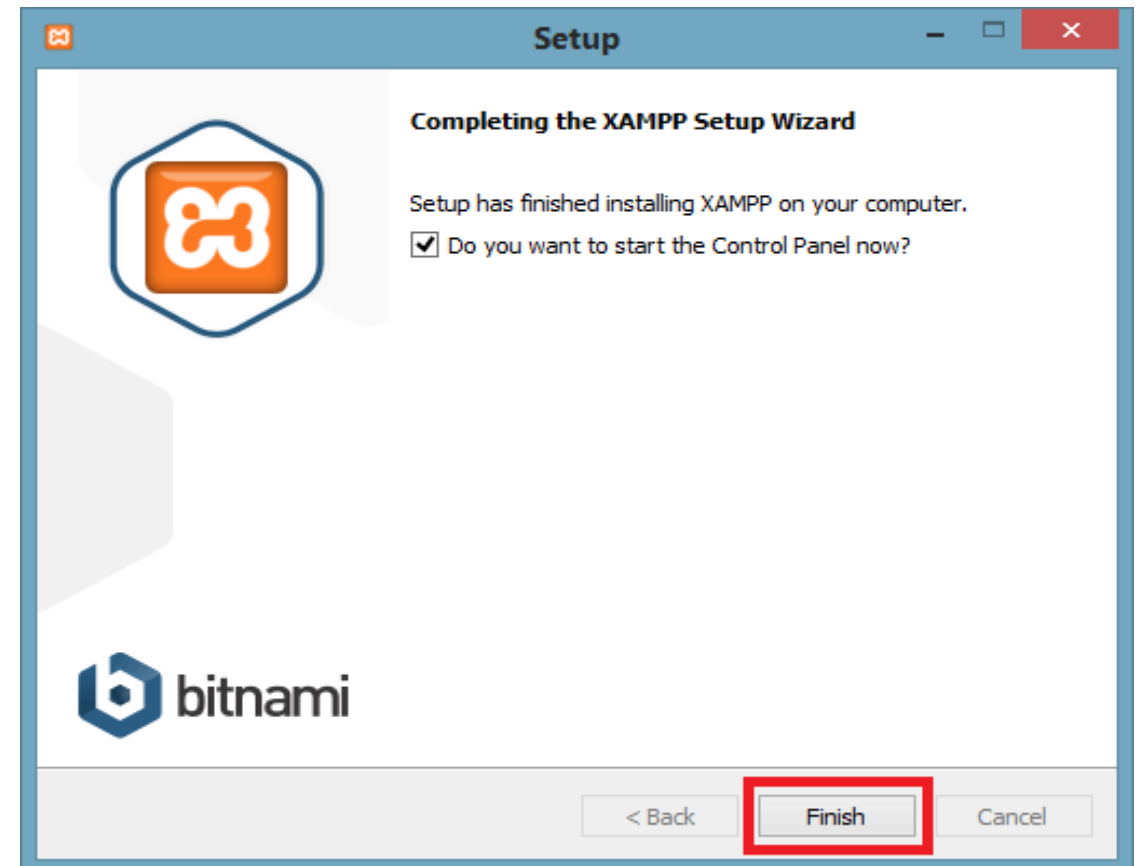
# Cont. ..



# Cont. ...



**Step 7:** A finish window will display after successful installation. Click on the Finish button.



## Cont. ...

**Step 8:** Choose your preferred language.

Language

  
☒

  
☐

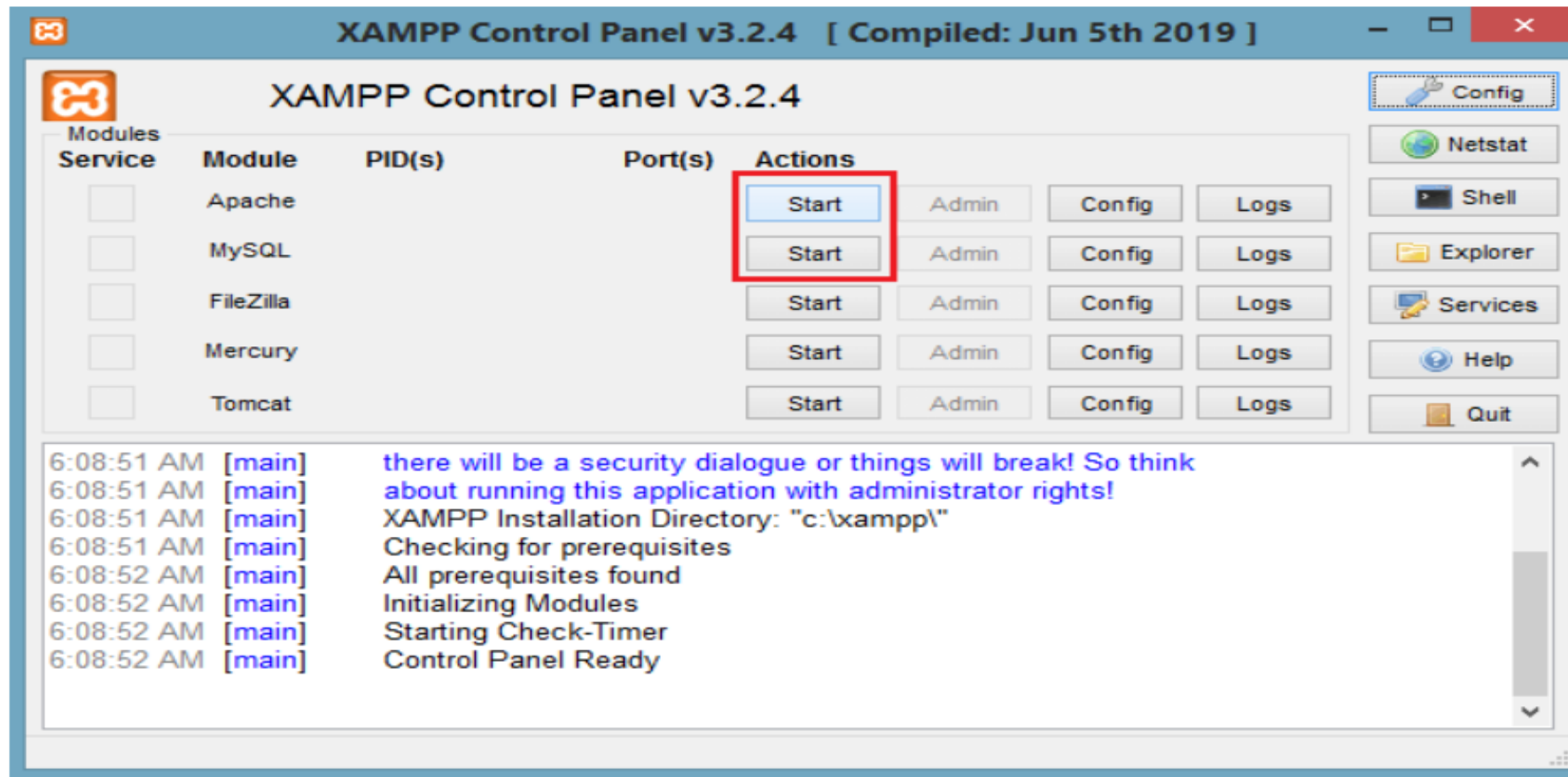
 Abort

 Save

# Cont. ...

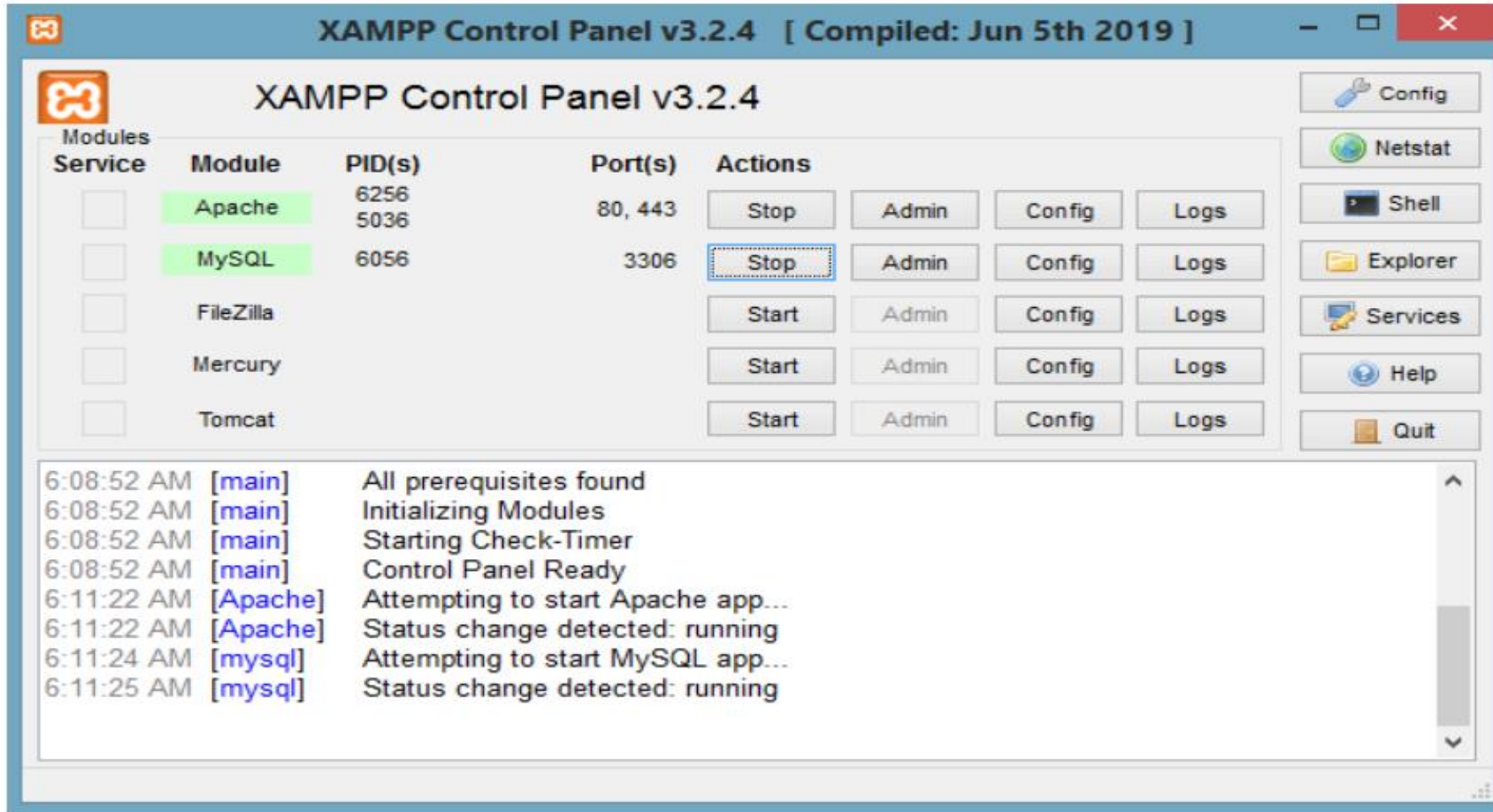
**Step 9:** XAMPP is ready to use. Start the Apache server and MySQL and run the php program on the localhost.

How to run PHP programs on XAMPP, see in the next tutorial.



# Cont. ...

**Step 10:** If no error is shown, then XAMPP is running successfully.



# 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
?>
```

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:



# Cont. ..

## Example:

php

 Copy code

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

<h1>Welcome to my PHP page</h1>
<?php
    echo "Hello, World!";
?>

</body>
</html>
```

# PHP Case Sensitivity



In PHP, case sensitivity plays a role in different parts of the language. Here's a breakdown of what is case-sensitive and what is not:

## 1. Case-Sensitive Elements


### a. Variable Names

PHP variable names are case-sensitive. This means that `$Name`, `$name`, and `$NAME` are treated as three different variables.

# Cont. ...

## Example:

php

 Copy code

```
<?php
$name = "Alice";
$name = "Bob";
echo $name; // Outputs: Alice
echo $name; // Outputs: Bob
?>
```

# Cont. ...


## 2. Case-Insensitive Elements

### a. Function Names

Function names in PHP are **case-insensitive**. For example, `echo()`, `ECHO()`, and `Echo()` will all work the same.

**Example:**

php

 Copy code

```
<?php
function greet() {
    echo "Hello!";
}

greet(); // Outputs: Hello!
GREET(); // Outputs: Hello!
GrEeT(); // Outputs: Hello!
?>
```




# Cont. ...

## b. Class Names

Class names are also **case-insensitive**. Once a class is declared, you can instantiate it with any capitalization.

**Example:**

php

 Copy code

```
<?php
class Car {
    public function drive() {
        echo "Driving!";
    }
}

$myCar = new Car();    // Works
$myCar = new CAR();    // Also works
$myCar = new car();    // Also works
?>
```




# Cont. ...

## c. Keywords and Language Constructs

PHP keywords like `if`, `else`, `while`, `for`, `echo`, etc., are case-insensitive.

### Example:

php

 Copy code

```
<?php
ECHO "This will work!"; // Outputs: This will work!
?>
```

### Summary:

- **Case-Sensitive:** Variables (`$myVar`  $\neq$  `$MyVar`).
- **Case-Insensitive:** Function names, class names, keywords, and language constructs (like `if`, `echo`, `while`).

# Comment in PHP



In PHP, comments are used to explain the code, make it more readable, or temporarily disable parts of it during debugging. PHP supports three types of comments:

## 1. Single-Line Comments

You can use `//` or `#` for single-line comments. Everything after `//` or `#` on the same line is treated as a comment.

**Example:**

php

Copy code

```
<?php
// This is a single-line comment using double slashes
echo "Hello, World!"; // This will print a message

# This is another single-line comment using a hash
echo "PHP comments are useful!";
?>
```




# Cont. ...

## 2. Multi-Line Comments

For multi-line comments, you can use the `/* */` syntax. This type of comment can span multiple lines.

**Example:**

php

 Copy code

```
<?php
/*
    This is a multi-line comment.
    It can span multiple lines.
    Use it for longer explanations.
*/
echo "Multi-line comments are useful for detailed notes.";
?>
```




# Cont. ...

## 3. Using Comments to Disable Code

You can also use comments to temporarily disable parts of your code during debugging.

**Example:**

php

 Copy code

```
<?php
echo "This will be executed.";

// echo "This line is commented out and won't be executed.";

# echo "This is also disabled with a hash.";
?>
```


# Multiple line of Comment



To write a multi-line comment in a single block of code in PHP, you can use the `/* */` syntax. Everything between `/*` and `*/` will be treated as a comment, allowing you to write across multiple lines.

## Example of a Multi-Line Comment in a Single Block:

php

 Copy code

```
<?php
/*
    This is a multi-line comment.
    You can write across multiple lines,
    and it will not be executed as code.
*/

echo "This line will be executed.";
?>
```



# PHP Variables



## PHP Variables

In PHP, variables are used to store data, which can be of various types such as strings, integers, arrays, etc. PHP variables are dynamically typed, meaning you don't have to declare their type explicitly.


### Key Features of PHP Variables:

1. **Starts with a \$ sign:** All variables in PHP must begin with a dollar sign ( \$ ), followed by the name of the variable.
2. **Must start with a letter or an underscore:** Variable names should start with a letter (A-Z or a-z) or an underscore ( \_ ), and subsequent characters can be letters, numbers, or underscores.
3. **Case-Sensitive:** Variable names are case-sensitive, meaning `$name` and `$Name` are different.
4. **No explicit type declaration:** PHP automatically determines the variable type based on the value assigned to it.

# Cont. ...

## Syntax

php

 Copy code


```
$variable_name = value;
```

## Examples

### 1. String Variables

A string is a sequence of characters, and you can assign a string to a variable.

php

 Copy code


```
<?php
$name = "John Doe"; // String variable
echo $name;         // Outputs: John Doe
?>
```

# Cont. ...

## 2. Integer Variables

An integer is a non-decimal number.

php


 Copy code

```
<?php
$age = 25; // Integer variable
echo $age; // Outputs: 25
?>
```

## 3. Float Variables

A float is a number with a decimal point or a number in exponential form.

php


 Copy code

```
<?php
$price = 19.99; // Float variable
echo $price;    // Outputs: 19.99
```

# Cont. ...

A boolean variable can hold only two values: `true` or `false`.

php


 Copy code

```
<?php
$is_admin = true;    // Boolean variable
echo $is_admin;      // Outputs: 1 (true is represented as 1)
?>
```

## 5. Array Variables

An array stores multiple values in one variable.

php

 Copy code

```
<?php
$fruits = array("Apple", "Banana", "Orange"); // Array variable
echo $fruits[0]; // Outputs: Apple
?>
```



## Cont. ...

### Variable Scope


PHP has three types of variable scope:

1. **Local:** Variables declared inside a function are local to that function.
2. **Global:** Variables declared outside a function can be accessed globally.
3. **Static:** Variables within a function retain their value after the function is executed.

# Cont. ...

## Example of Local and Global Scope:

php

 Copy code

```
<?php
$globalVar = "Global"; // Global variable

function test() {
    $localVar = "Local"; // Local variable
    echo $localVar;       // Outputs: Local
}


test();
echo $globalVar;         // Outputs: Global
?>
```



# Cont. ...

## Example of Static Variables:

php

 Copy code

```
<?php
function counter() {
    static $count = 0; // Static variable
    $count++;
    echo $count;
}

counter(); // Outputs: 1
counter(); // Outputs: 2
?>
```


# Cont. ...

## Concatenating Variables

You can concatenate (combine) strings and variables using the dot ( `.` ) operator.

Example:

php

 Copy code

```
<?php
$first_name = "John";
$last_name = "Doe";
echo $first_name . " " . $last_name; // Outputs: John Doe
?>
```


# Cont. ...

## Variable Variables

PHP allows the creation of **variable variables**, meaning the value of a variable can become the name of another variable.

**Example:**

php

 Copy code

```
<?php
$var = "name";
$$var = "John"; // Creates a variable $name with the value "John"
echo $name;      // Outputs: John
?>
```


# Cont. ...

## Superglobal Variables

PHP provides several built-in superglobals which are always accessible, regardless of scope. Some examples include `$_GET`, `$_POST`, `$_SERVER`, etc.

Example of Superglobal Variable:

php

 Copy code

```
<?php
echo $_SERVER['SERVER_NAME']; // Outputs the name of the server
?>
```

# PHP echo and print Statements



In PHP, both `echo` and `print` are used to output data to the screen. While they are similar in many ways, there are a few subtle differences between the two.


## 1. `echo` Statement

- **Faster:** Slightly faster than `print`, because `echo` does not return a value.
- **No Return Value:** It doesn't return anything, making it a better choice when simply outputting data.
- **Multiple Parameters:** Can accept multiple parameters (though using commas between parameters is rarely used in practice).

# Cont. ...

## Syntax:


php

 Copy code

```
echo expression1, expression2, ...;
```

## Example:

php

 Copy code

```
<?php  
echo "Hello, World!"; // Outputs: Hello, World!  
echo "I am ", "learning ", "PHP"; // Outputs: I am learning PHP  
?>
```


# Cont. ...

## 2. `print` Statement

- **Slower:** Slightly slower than `echo`, because it always returns a value ( `1` ).
- **Returns a Value:** `print` returns `1`, so it can be used in expressions.
- **Single Parameter:** Only accepts a single parameter.

Syntax:

php


 Copy code

```
print(expression);
```

# Cont. ...

## Example:

php

 Copy code

```
<?php
print "Hello, World!"; // Outputs: Hello, World!

$success = print "Hello, PHP"; // Outputs: Hello, PHP
echo $success; // Outputs: 1, because print returns 1
?>
```



# Cont. ...


## Key Differences between `echo` and `print`

Feature	<code>echo</code>	<code>print</code>
Return Value	No return value	Returns <code>1</code> (can be used in expressions)
Speed	Slightly faster	Slightly slower
Number of Parameters	Can take multiple parameters	Can only take one parameter
Usage in Expressions	Cannot be used in expressions	Can be used in expressions

## Examples Highlighting the Differences:

### Example 1: Multiple Parameters with `echo`

php

 Copy code


```
<?php
echo "Learning", " PHP", " is", " fun!"; // Outputs: Learning PHP is fun!
?>
```



# Cont. ...

## Example 2: `print` in Expressions


php

 Copy code

```
<?php
$print_result = print "Hello, PHP!"; // Outputs: Hello, PHP!
echo $print_result; // Outputs: 1 (because print returns 1)
?>
```

## Example: Using Both in the Same Script

php

 Copy code

```
<?php
echo "This is an echo statement."; // Outputs: This is an echo statement.
print " This is a print statement."; // Outputs: This is a print statement.
?>
```



# Thank you!

Appreciate your action.