PHP PROGRAMMING

Unit - 1

Introduction to PHP

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf released the first version of PHP way back in 1994. Initially, PHP was supposed to be an abbreviation for "Personal Home Page", but it now stands for the recursive initialism "PHP: Hypertext Preprocessor".

Lerdorf began PHP development in 1993 by writing several Common Gateway Interface (CGI) programs in C, which he used to maintain in his personal homepage. Later on, He extended them to work with web forms and to communicate with databases. This implementation of PHP was "Personal Home Page/Forms Interpreter" or PHP/FI.

Today, PHP is the world’s most popular server-side programming language for building web applications. Over the years, it has gone through successive revisions and versions.

Here are some more important features of PHP −

- PHP performs system functions. It can create, open, read, write, and close the files.
- PHP can handle forms. It can gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

PHP provides a large number of reusable classes and libraries are available on "PEAR" and "Composer". PEAR (PHP Extension and Application Repository) is a distribution system for reusable PHP libraries or classes. "Composer" is a dependency management tool in PHP.

Basic Knowledge of websites

A website is a collection of many web pages, and web pages are digital files that are written using HTML(HyperText Markup Language). To make your website available to every person in the world, it must be stored or hosted on a computer connected to the Internet round a clock. Such computers are known as a **Web Server**. The website’s web pages are linked with hyperlinks and hypertext and share a common interface and design. The website might also contain some additional documents and files such as images, videos, or other digital assets.

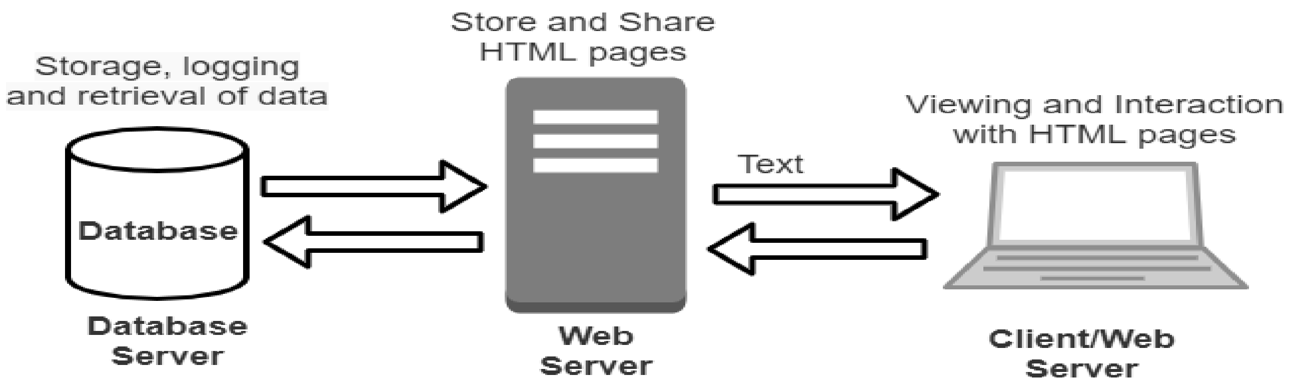
Components of a Website: We know that a website is a collection of a webpages hosted on a web-server. These are the components for making a website.

- **Webhost:** Hosting is the location where the website is physically located. Group of webpages (linked webpages) licensed to be called a website only when the webpage is hosted on the webserver. The webserver is a set of files transmitted to user computers when they specify the website's address..
- **Address:** Address of a website also known as the URL of a website. When a user wants to open a website then they need to put the address or URL of the website into the web browser, and the asked website is delivered by the webserver.
- **Homepage :** Home page is a very common and important part of a webpage. It is the first webpage that appears when a visitor visits the website. The home page of a website is very important as it sets the look and feel of the website and directs viewers to the rest of the pages on the website.
- **Design :** It is the final and overall look and feel of the website that has a result of proper use and integration elements like navigation menus, graphics, layout, navigation menus etc.
- **Content :** Every web pages contained on the website together make up the content of the website. Good content on the webpages makes the website more effective and attractive.
- **The Navigation Structure:** The navigation structure of a website is the order of the pages, the collection of what links to what. Usually, it is held together by at least one navigation menu.

DYNAMIC WEBSITE

Dynamic Website is a website containing data that can be mutable or changeable. It uses client-side or server scripting to generate mutable content. Like a static website, it also contains HTML data.
Dynamic websites are those websites that changes the content or layout with every request to the webserver. These websites have the capability of producing different content for different visitors from the same source code file. There are two kinds of dynamic web pages i.e. client side scripting and server side scripting. The client-side web pages changes according to your activity on the web page. On the server-side, web pages are changed whenever a web page is loaded.

Example: login & signup pages, application & submission forms, inquiry and shopping cart pages.
A Typical Architecture of dynamic website


Storage, logging and retrieval of data
Database
Database Server
Store and Share HTML pages
Web Server
Text
Viewing and Interaction with HTML pages
Client/Web Server

There are different languages used to create dynamic web pages like PHP, ASP, .NET and JSP. Whenever a dynamic page loads in browser, it requests the database to give information depending upon user's input. On receiving information from the database, the resulting web page is applied to the user after applying the styling codes.

Features of dynamic webpage:

- These websites are very flexible.
- In these websites the content can be quickly changed on the user's computer without new page request to the web browser.
- In these websites the owner have the ability to simply update and add new content to the site.
- These websites are featured with content management system, e-commerce system and intranet or extranet facilities.
- Most of the dynamic web content, is assembled on the web using server-scripting languages.

Here is a list of application where we use dynamic website in real world.

- Facebook
- Twitter
- LinkedIn
- Online booking website.
- Social media

SCOPE OF PHP

- **Dynamic Typing:** PHP is dynamically typed, meaning you don't need to declare the data type of a variable explicitly.
- **Cross-Platform:** PHP runs on various platforms, making it compatible with different operating systems.
- **Database Integration:** PHP provides built-in support for interacting with databases, such as MySQL, PostgreSQL, and others.
- **Server-Side Scripting:** PHP scripts are executed on the server, generating HTML that is sent to the client's browser.

PHP is versatile and can be used in a variety of web development scenarios, including:

- **Dynamic Web Pages**: Generating dynamic content based on user interaction or other conditions.
- **Content Management Systems (CMS):** Many popular CMSs like WordPress, Joomla, and Drupal are built with PHP.
- **E-commerce Platforms:** PHP is commonly used to develop e-commerce websites due to its database integration capabilities.
- **Web Applications:** PHP is used for creating feature-rich web applications such as social media platforms, forums, and customer relationship management (CRM) systems.
- **API Development:** PHP can be used to create APIs for web and mobile applications.

XAMPP and WAMP Installation

WAMP

WAMP is an acronym for Windows, Apache, MySQL, and PHP, it's used as a Web development platform that insists the users manage web apps using PHP, MySQL, and Apache. It is free and an open-source web server solution stack package. It uses virtual hosts and provides top services.

NOTE: WAMP IS DESIGNED ONLY FOR WINDOWS, IT CAN NOT BE INSTALLED ON MAC OS AND LINUX

XAMPP

XAMPP is an acronym for (X) cross-platform (Windows, Linux, Mac OS), (A)Apache, (M)Maria DB or MySQL, (P)PHP and Perl. Like WAMP, XAMPP is also a free and open-source platform used for web development.

NOTE: UNLIKE WAMP, XAMPP IS DESIGNED FOR WINDOWS, MAC OS AND LINUX. SO IT CAN BE INSTALLED IN ALL OPERATING SYSTEMS

The installation process in Windows

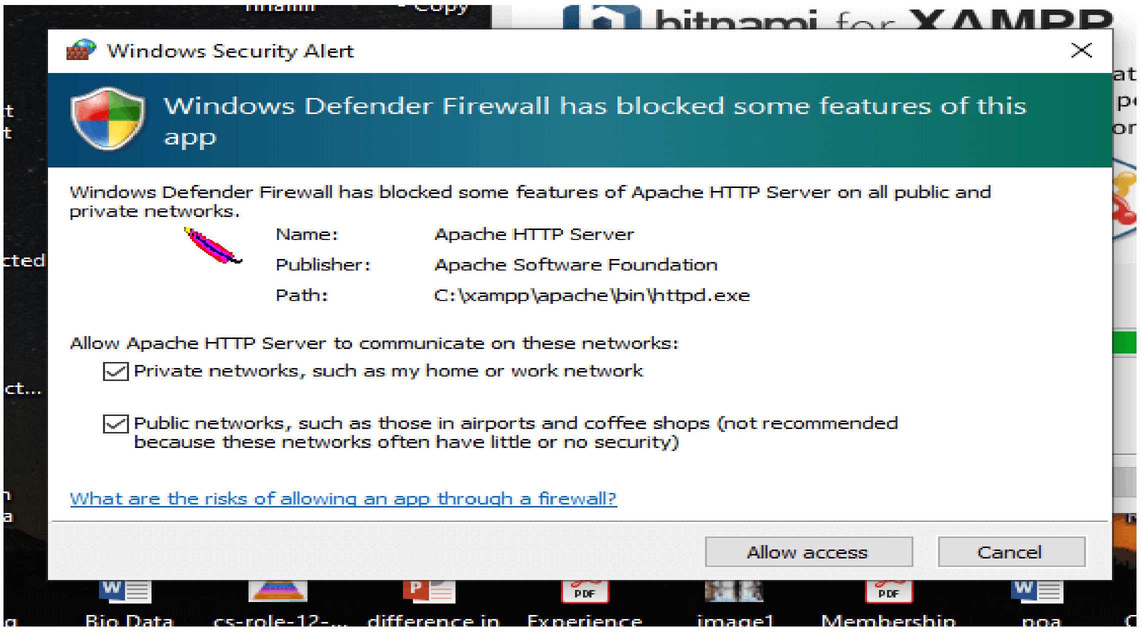
STEP 1- Open any web browser and visit https://www.apachefriends.org/index.html. On the home page, you can find the option to download XAMPP for three platforms- Windows, MAC, and Linux. Click on **XAMPP for Windows**. The latest version available on the website is **7.4.5.**

As soon as you click on it, a message displaying the automatic start of download appears on the screen.

STEP 2- After the download is completed, double click the .exe extension file to start the process of installation.

STEP 3- A pop-up screen with the message asking you to allow to make changes on your desktop appears. Click "YES" to continue the process.

STEP 4- Click to **Allow access** or deactivate the firewall and any other antivirus software because it can hamper the process of installation. Thus, it is required to temporarily disable any antivirus software or security firewall till the time all the XAMPP components have been installed completely.



Windows Security Alert

Windows Defender Firewall has blocked some features of this app

Windows Defender Firewall has blocked some features of Apache HTTP Server on all public and private networks.

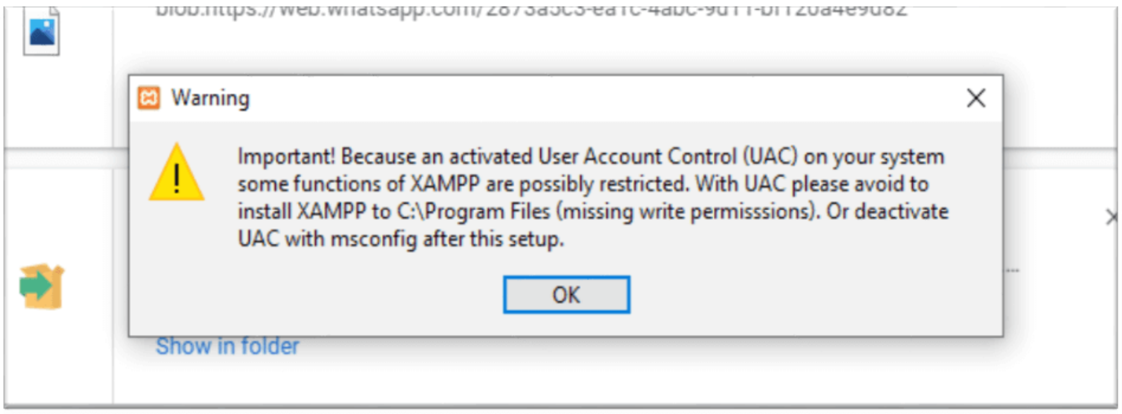
Name: Apache HTTP Server
Publisher: Apache Software Foundation
Path: C:\xampp\apache\bin\httpd.exe

Allow Apache HTTP Server to communicate on these networks:

- [✓] Private networks, such as my home or work network
- [✓] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

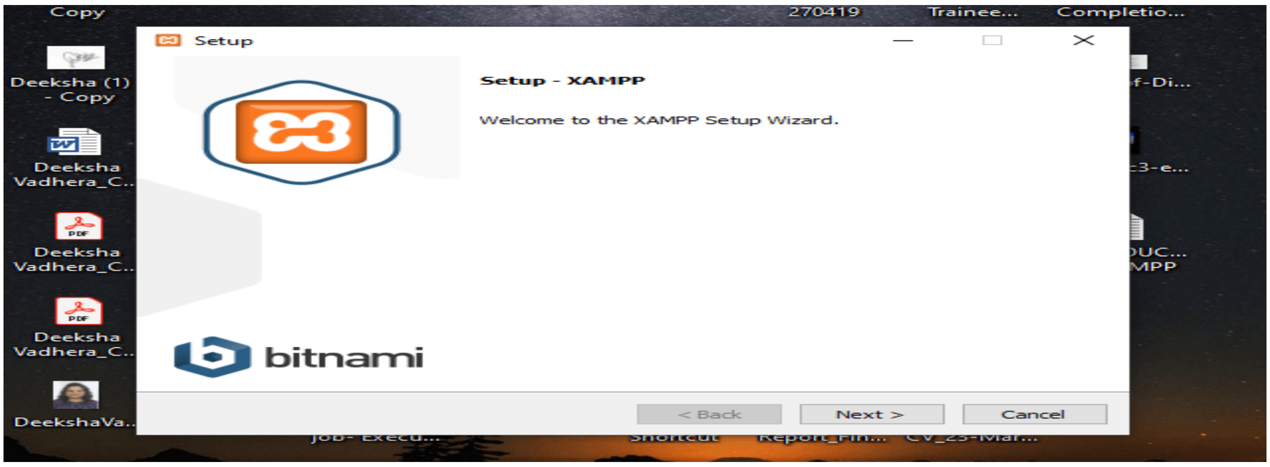
STEP 5- Just before the installation, a pop-up window appears with a warning to **disable UAC**. User Account Control (UAC) interrupts the XAMPP installation because it restricts the access to write to the C: drive. Therefore, it is suggested to disable it for the period of installation.



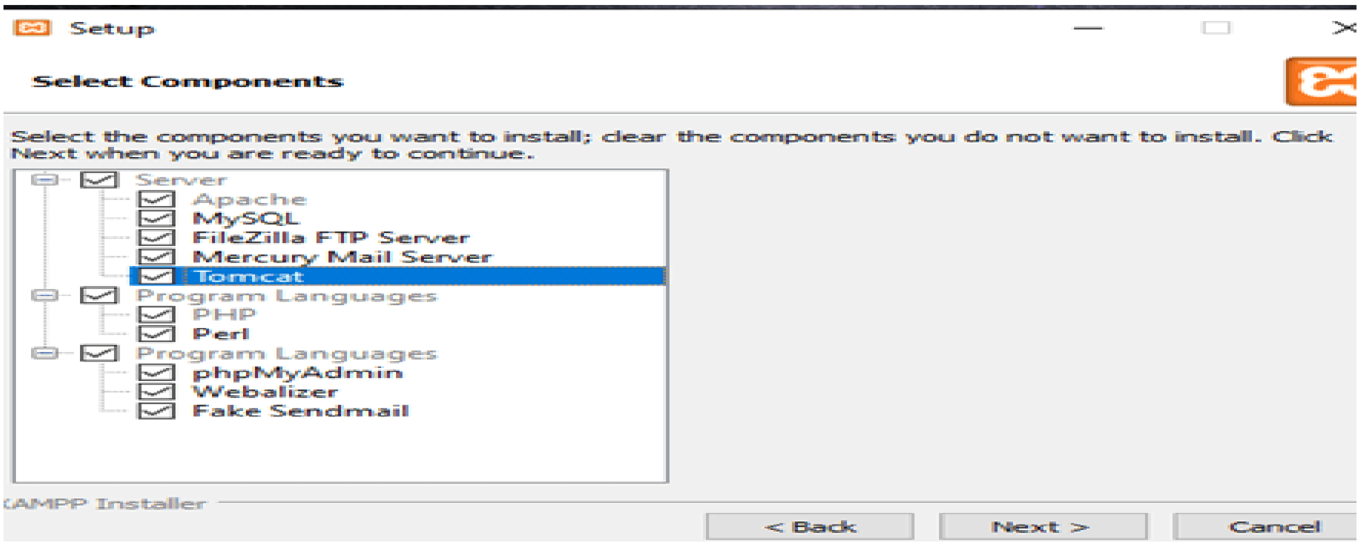
Warning

Important! Because an activated User Account Control (UAC) on your system some functions of XAMPP are possibly restricted. With UAC please avoid to install XAMPP to C:\Program Files (missing write permisssions). Or deactivate UAC with msconfig after this setup.

STEP 6- After clicking the .exe extension file, the XAMPP setup wizard opens spontaneously. Click on "NEXT" to start the configuration of the settings.



STEP 7- After that, a 'Select Components' panel appears, which gives you the liberty to choose amongst the separate components of the XAMPP software stack for the installation. To get a complete localhost server, it is recommended to install using the default options of containing all available components. Click "NEXT" to proceed further.

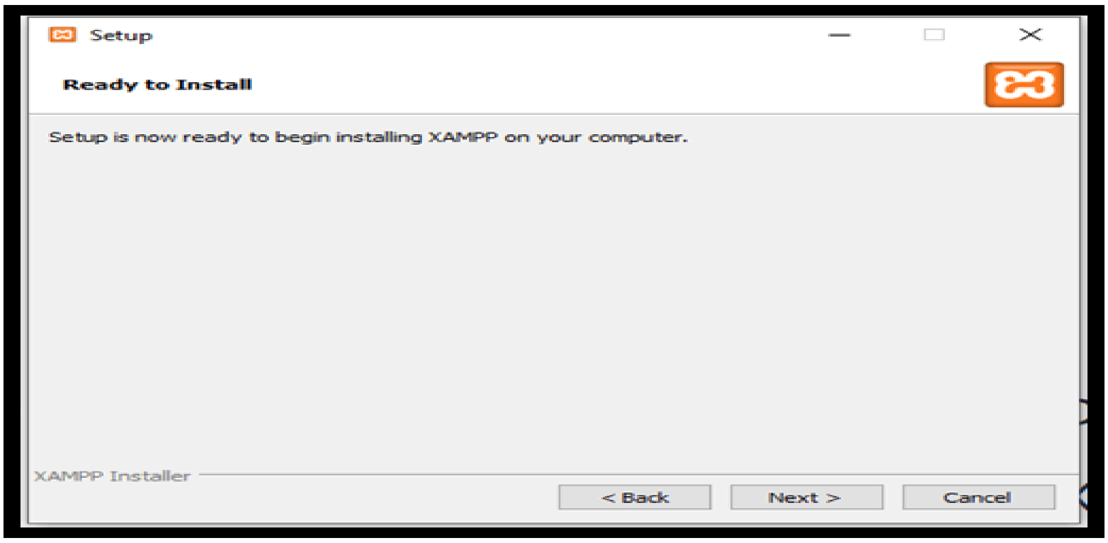

Setup

Select Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer
  - ☑ Fake Sendmail

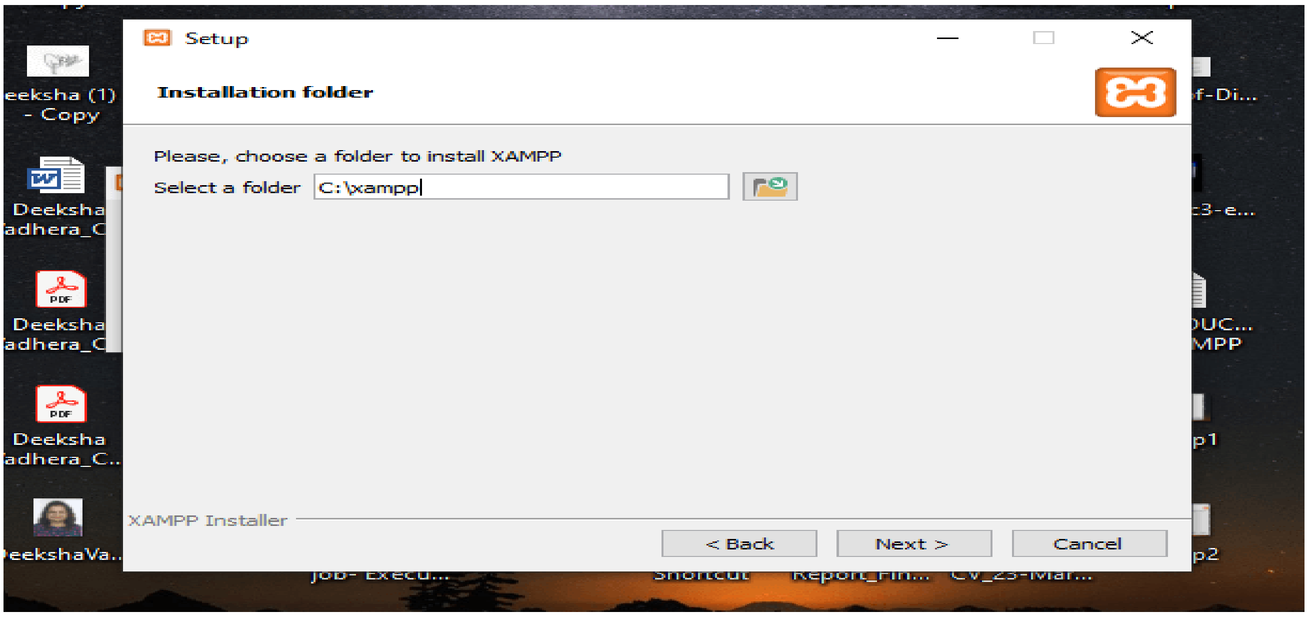
STEP 8- The setup is now ready to install, and a pop-up window showing the same appears on the screen. Click "NEXT" to take the process forward.


Setup

Ready to Install

Setup is now ready to begin installing XAMPP on your computer.

STEP 9- Select the location where the XAMPP software packet needs to be installed. The original setup creates a folder titled XAMPP under C:\ for you. After choosing a location, click "NEXT".


Setup

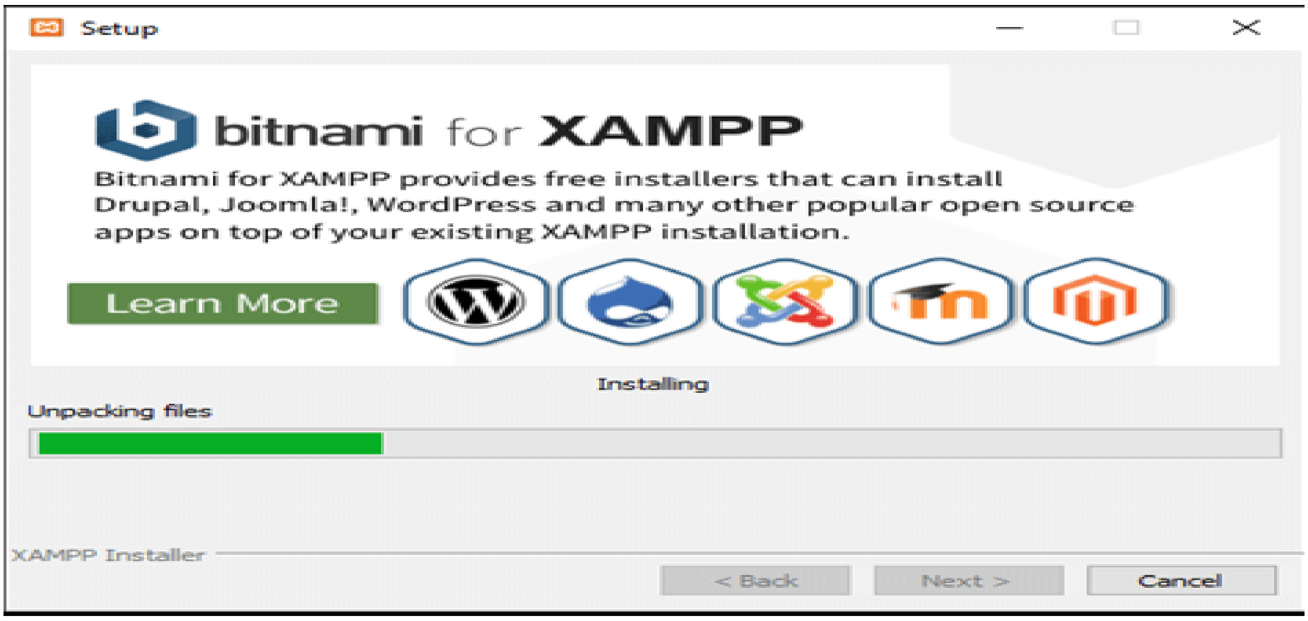
Installation folder

Please, choose a folder to install XAMPP

Select a folder C:\xampp

XAMPP Installer

STEP 10- After choosing from all the previously mentioned preferences (like language and learn more bitnami dialogue box) click to start the installation. The setup wizard will unpack and install the components to your system. The components are saved to the assigned directory. This process may takes a few minutes to complete. The progress of the installation in terms of percentage is visible on the screen.


Setup

bitnami for XAMPP

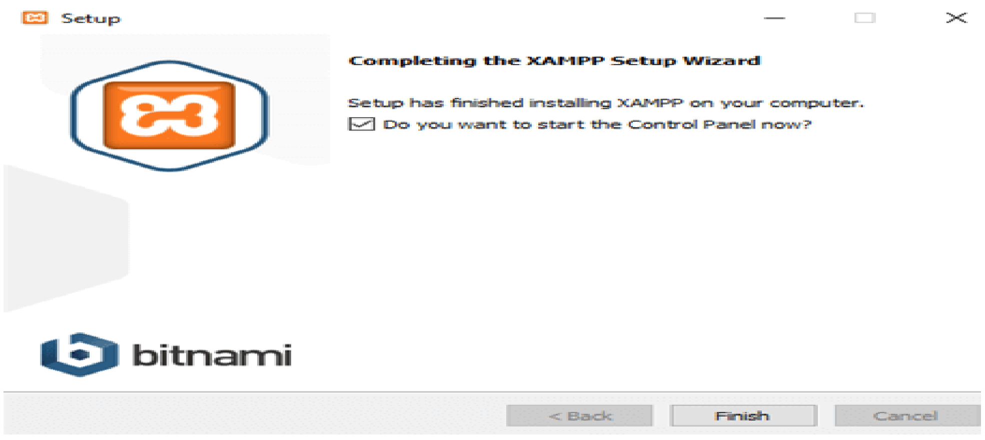
Bitnami for XAMPP provides free installers that can install Drupal, Joomla!, WordPress and many other popular open source apps on top of your existing XAMPP installation.

Installing

Unpacking files

XAMPP Installer

STEP 11- After the successful installation of the XAMPP setup on your desktop, press the "FINISH" button.


Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

☑ Do you want to start the Control Panel now?

On clicking the FINISH button, the software automatically launches, and the CONTROL PANEL is visible. The image below shows the appearance of the final result.

UNIT – 2

PHP PROGRAMMING BASICS

Syntax of PHP

PHP, a powerful server-side scripting language used in web development. It's simplicity and ease of use makes it an ideal choice for beginners and experienced developers. This article provides an overview of PHP syntax. PHP scripts can be written anywhere in the document within PHP tags along with normal HTML.

Basic PHP Syntax

PHP code is executed between PHP tags, allowing the integration of PHP code within HTML. The most common PHP tag is **<?php … ?>**, which is used to enclose PHP code. The **<?php ….?>** is called **Escaping to PHP**.

```
<?php
 // code
?>
```

The script starts with **<?php** and ends with **?>**. These tags are also called 'Canonical PHP tags'. Everything outside of a pair of opening and closing tags is ignored by the PHP parser. The open and closing tags are called delimiters. Every PHP command ends with a semi-colon (**;**).

Basic Example of PHP

```
<?php
// Here echo command is used to print
echo "Hello, world!";
?>
```

Output

```
Hello, world!
```

Embedding PHP in HTML

PHP code can be embedded within HTML using the standard PHP tags. In this example, the <?php echo “Hello, PHP!”; ?> statement dynamically inserts a heading into the HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>PHP Syntax Example</title>
</head>
<body>
   <h1><?php echo "Hello, PHP!"; ?></h1>
</body>
</html>
```

SGML or Short HTML Tags

These are the shortest option to initialize a PHP code. The script starts with **<?** and ends with **?>**. This will only work by setting the *short_open_tag* setting in the *php.ini* file to ‘on’.

Example:

```
<?
// Here echo command will only work if
// setting is done as said before
echo "Hello, world!";
?>
```

Output

```
Hello, world!
```

Embedding HTML in PHP

PHP is a server-side scripting language. We need to handle some logic by server-side in PHP web applications, allowing developers to embed HTML directly within PHP code and generate dynamic content for web applications.

The following ways to embed HTML in PHP are:

- Basic Embedding
- Using Short Tags
- Embedding HTML Blocks
- PHP Templating Engines

Basic Embedding

The simplest way to embed HTML in PHP is by mixing PHP tags () with HTML code. Let's take a simple example:

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>PHP HTML Integration</title> </head> <body> <h1><?php echo "Welcome to PHP HTML Integration"; ?></h1> <p><?php echo "This is a basic example of embedding HTML in PHP."; ?></p> </body> </html>
```

In the above code, We have encapsulated dynamic content in PHP tags. The echo statement outputs text or variables to the HTML.

Using Short Tags

PHP also supports short tags () to embed HTML in PHP. This provides a more concise way to embed HTML. Some servers may have short tags disabled for security or compatibility reasons. Please verify and enable it on the server.

```
<!DOCTYPE html> <html lang="en">
<head>
```

```
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>PHP HTML Integration</title>
</head>
<body>
<h1>
<?= "Welcome to PHP HTML Integration"; ?>
</h1>
<p>
<?= "This is another example using short tags."; ?>
</p>
 </body>
</html>
```

The shorthand achieves the same result as `<?PHP ?>`.

Embedding HTML Blocks

we can also use **heredoc** or **nowdoc** syntax to embed larger HTML blocks or templates. Encapsulating HTML within PHP variables enables a more organized structure of code. You can include longer HTML blocks with **heredoc** and **nowdoc** without having to perform a lot of concatenation.

Heredoc Syntax:

```
<?php $htmlContent = <<<HTML <h1>Welcome to Heredoc PHP HTML</h1> <p>This is a more extensive example using heredoc syntax.</p> HTML; echo $htmlContent; ?>
```

Nowdoc Syntax:

```
<?php $htmlContent = <<<'HTML' <h1>Welcome to Nowdoc PHP HTML</h1> <p>This is a more extensive example using nowdoc syntax.</p> HTML; echo $htmlContent; ?>
```

PHP Templating Engines

The Template Engine is the most popular way to integrate HTML into PHP for complex web applications. This helps to create a clean separation of HTML presentation and PHP logic in the web application, making it easier to manage and maintain code.

Simple Example of Blade Template:

```
<!DOCTYPE html>
 <html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Blade Templating Example</title>
 </head>
 <body>
<h1>{{ "Welcome to Blade Templating" }}</h1>
 </body>
</html>
```

Introduction to PHP Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

1. $variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).

- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

Output:

```
string is: hello string
integer is: 200
float is: 44.6
```

PHP Variable: Sum of two variables

File: variable2.php

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Output:

```
11
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

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

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5
My boat is
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>
```

Output:

```
hello
hello
```

File: variableinvalid.php

```
<?php
$4c="hello";//number (invalid)
```

```
3. $*d="hello";//special symbol (invalid)
4. echo "$4c <br/> $*d";
5. ?>
```

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable (T_VARIABLE)
or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.

Understanding Data Types in PHP

PHP data types are a fundamental concept to defines how variables store and manipulate data. PHP is a loosely typed language, which means variables do not need to be declared with a specific data type. PHP allows eight different types of data types. All of them are discussed below.

Table of Content

- Predefined Data Types
  - Integer
  - Float (Double)
  - String
  - Boolean
- User-Defined (compound) Data Types
  - Array
  - Objects
- Special Data Types
  - NULL
  - Resources

Predefined Data Types

1. Integer

Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8), or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^31 to 2^31.

```
<?php
// decimal base integers
$deci1 = 50;
$deci2 = 654;
// octal base integers
$octal1 = 07;
// hexadecimal base integers
$octal = 0x45;
$sum = $deci1 + $deci2;
echo $sum;
echo "\n\n";
//returns data type and value
var_dump($sum)
?>
```

Output

```
704

int(704)
```

2. Float (Double)

Can hold numbers containing fractional or decimal parts including positive and negative numbers or a number in exponential form. By default, the variables add a minimum number of decimal places. The Double data type is the same as a float as floating-point numbers or real numbers.

```
<?php
$val1 = 50.85;
$val2 = 654.26;
$sum = $val1 + $val2;
echo $sum;
echo "\n\n";
//returns data type and value
var_dump($sum)
?>
```

Output

```
705.11
```

```
float(705.11)
```

3. String

Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes, but they will be treated differently while printing variables. To clarify this look at the example below.

```
<?php
$name = "Krishna";
echo "The name of the Geek is $name \n";
echo 'The name of the geek is $name ';
echo "\n\n";
//returns data type, size and value
var_dump($name)
?>
```

Output

```
The name of the Geek is Krishna

The name of the geek is $name

string(7) "Krishna"
```

4. Boolean

Boolean data types are used in conditional testing. Hold only two values, either TRUE(1) or FALSE(0). Successful events will return *true* and unsuccessful events return *false*. NULL type values are also treated as *false* in Boolean. Apart from NULL, 0 is also considered false in boolean. If a string is empty then it is also considered false in boolean data type.

```
<?php
if(TRUE)
   echo "This condition is TRUE";
if(FALSE)
   echo "This condition is not TRUE";
?>
```

Output

```
This condition is TRUE
```

User-Defined (compound) Data Types

1. Array

Array is a compound data type that can store multiple values of the same data type. Below is an example of an array of integers. It combines a series of data that are related together.

```
<?php
$intArray = array( 10, 20 , 30);
echo "First Element: $intArray[0]\n";
echo "Second Element: $intArray[1]\n";
echo "Third Element: $intArray[2]\n\n";
//returns data type and value
var_dump($intArray);
?>
```

Output

```
First Element: 10

Second Element: 20

Third Element: 30

array(3) {

  [0]=>

  int(10)

  [1]=>

  int(20)

  [2]=>

  int(30)

}
```

We will discuss arrays in detail in further articles.

2. Objects

Objects are defined as instances of user-defined classes that can hold both values and functions and information for data processing specific to the class. This is an advanced topic and will be discussed in detail in further articles. When the objects are created, they inherit all the properties and behaviours from the class, having different values for all the properties.

Objects are explicitly declared and created from the *new* keyword.

```
<?php
class gfg {
      public $message;
        function __construct($message) {
        $this->message = $message;
      }
      function msg() {
        return "This is an example of " . $this->message . "!";
      }
}
$newObj = new gfg("Object Data Type");
echo $newObj->msg();
?>
```

Output

```
This is an example of Object Data Type!
```

Special Data Types

1. NULL

These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but it’s case-insensitive NULL, Null, and null are treated the same. If a variable is created without a value or no value, it is automatically assigned a value of NULL. It is written in capital letters.

```
<?php
$nm = NULL;
echo $nm;    // this will return no output
// return data type
var_dump($nm);
?>
```

Output

```
NULL
```

2. Resources

Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource. Resource variables hold special handles for files and database connections.

Note:

- To check the type and value of an expression, use the **var_dump()** function which dumps information about a variable.
- PHP allows the developer to cast the data type.

USING OPERATORS IN PHP

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values.

PHP Operators can be categorized in following forms:

- Arithmetic Operators
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators
- Execution Operators
- Error Control Operators

Arithmetic Operators:

The arithmetic operators are used to perform simple mathematical operations like addition, subtraction, multiplication, etc. Below is the list of arithmetic operators along with their syntax and operations in PHP.

Operator	***Name***	***Syntax***	***Operation***
+	Addition	$x + $y	Sum the operands
–	Subtraction	$x – $y	Differences the operands
*	Multiplication	$x * $y	Product of the operands

Operator	***Name***	***Syntax***	***Operation***
/	Division	$x / $y	The quotient of the operands
**	Exponentiation	$x ** $y	$x raised to the power $y
%	Modulus	$x % $y	The remainder of the operands

Logical or Relational Operators:

These are basically used to operate with conditional statements and expressions. Conditional statements are based on conditions. Also, a condition can either be met or cannot be met so the result of a conditional statement can either be true or false. Here are the logical operators along with their syntax and operations in PHP.

Operator	***Name***	***Syntax***	***Operation***
and	Logical AND	$x and $y	True if both the operands are true else false
or	Logical OR	$x or $y	True if either of the operands is true else false
xor	Logical XOR	$x xor $y	True if either of the operands is true and false if both are true
&&	Logical AND	$x && $y	True if both the operands are true else false
||	Logical OR	$x || $y	True if either of the operands is true else false
!	Logical NOT	!$x	True if $x is false

Comparison Operators: These operators are used to compare two elements and outputs the result in boolean form. Here are the comparison operators along with their syntax and operations in PHP.

Operator	Name	Syntax	Operation
==	Equal To	$x == $y	Returns True if both the operands are equal
!=	Not Equal To	$x != $y	Returns True if both the operands are not equal
<>	Not Equal To	$x <> $y	Returns True if both the operands are unequal
===	Identical	$x === $y	Returns True if both the operands are equal and are of the same type
!==	Not Identical	$x == $y	Returns True if both the operands are unequal and are of different types
<	Less Than	$x < $y	Returns True if $x is less than $y
>	Greater Than	$x > $y	Returns True if $x is greater than $y
<=	Less Than or Equal To	$x <= $y	Returns True if $x is less than or equal to $y
>=	Greater Than or Equal To	$x >= $y	Returns True if $x is greater than or equal to $y

Conditional or Ternary Operators:

These operators are used to compare two values and take either of the results simultaneously, depending on whether the outcome is TRUE or FALSE. These are also used as a shorthand notation for *if…else* statement that we will read in the article on decision making.

Syntax:

```
$var = (condition)? value1 : value2;
```

Here, the condition will either evaluate as true or false. If the condition evaluates to True, then value1 will be assigned to the variable $var otherwise value2 will be assigned to it.

Operator	***Name***	***Operation***
?:	Ternary	If the condition is true? then $x : or else $y. This means that if the condition is true then the left result of the colon is accepted otherwise the result is on right

Assignment Operators:

These operators are used to assign values to different variables, with or without mid-operations. Here are the assignment operators along with their syntax and operations, that PHP provides for the operations.

Operator	***Name***	***Syntax***	***Operation***
=	Assign	$x = $y	Operand on the left obtains the value of the operand on the right
+=	Add then Assign	$x += $y	Simple Addition same as $x = $x + $y
-=	Subtract then Assign	$x -= $y	Simple subtraction same as $x = $x – $y
*=	Multiply then Assign	$x *= $y	Simple product same as $x = $x * $y
/=	Divide then Assign (quotient)	$x /= $y	Simple division same as $x = $x / $y
%=	Divide then Assign (remainder)	$x %= $y	Simple division same as $x = $x % $y

Array Operators:

These operators are used in the case of arrays. Here are the array operators along with their syntax and operations, that PHP provides for the array operation.

Operator	***Name***	***Syntax***	***Operation***
+	Union	$x + $y	Union of both i.e., $x and $y
==	Equality	$x == $y	Returns true if both has same key-value pair
!=	Inequality	$x != $y	Returns True if both are unequal
===	Identity	$x === $y	Returns True if both have the same key-value pair in the same order and of the same type
!==	Non-Identity	$x !== $y	Returns True if both are not identical to each other
<>	Inequality	$x <> $y	Returns True if both are unequal

Increment/Decrement Operators:

These are called the unary operators as they work on single operands. These are used to increment or decrement values.

Operator	***Name***	***Syntax***	***Operation***
++	Pre-Increment	++$x	First increments $x by one, then return $x
—	Pre-Decrement	–$x	First decrements $x by one, then return $x
++	Post-Increment	$x++	First returns $x, then increment it by one
—	Post-Decrement	$x–	First returns $x, then decrement it by one

String Operators:

This operator is used for the concatenation of 2 or more strings using the concatenation operator (‘.’). We can also use the concatenating assignment operator (‘.=’) to append the argument on the right side to the argument on the left side.

Operator	***Name***	***Syntax***	***Operation***
.	Concatenation	$x.$y	Concatenated $x and $y
.=	Concatenation and assignment	$x.=$y	First concatenates then assigns, same as $x = $x.$y

Spaceship Operators:

PHP 7 has introduced a new kind of operator called spaceship operator. The spaceship operator or combined comparison operator is denoted by “<=>“. These operators are used to compare values but instead of returning the boolean results, it returns integer values. If both the operands are equal, it returns 0. If the right operand is greater, it returns -1. If the left operand is greater, it returns 1. The following table shows how it works in detail:

Operator	***Syntax***	***Operation***
$x < $y	$x <=> $y	Identical to -1 (right is greater)
$x > $y	$x <=> $y	Identical to 1 (left is greater)
$x <= $y	$x <=> $y	Identical to -1 (right is greater) or identical to 0 (if both are equal)
$x >= $y	$x <=> $y	Identical to 1 (if left is greater) or identical to 0 (if both are equal)
$x == $y	$x <=> $y	Identical to 0 (both are equal)
$x != $y	$x <=> $y	Not Identical to 0

USING CONDITIONAL STATEMENT IN PHP

PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two- way path. If you want something then go this way or else turn that way. To use this feature, PHP provides us with four conditional statements:

- **if** statement
- **if…else** statement
- **if…elseif…else** statement
- **switch** statement

Let us now look at each one of these in details:

if Statement: This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.

Syntax :

```
if (condition){
   // if TRUE then execute this code
}
```

Example:

```
<?php
$x = 12;
if ($x > 0) {
        echo "The number is positive";
}
?>
```

Output:

```
The number is positive
```

if…else Statement: We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
else{
   // if FALSE then execute this code
}
```

Example :

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

Output:

12 is even number

<u>if…elseif…else Statement</u>: This allows us to use multiple if…else statements. We use this when there are multiple conditions of TRUE cases.

Syntax:

```
if (condition) {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
elseif {
   // if TRUE then execute this code
}
else {
   // if FALSE then execute this code
}
```

Example:

```
<?php
$x = "August";
if ($x == "January") {
        echo "Happy Republic Day";
}
```

```
elseif ($x == "August") {
        echo "Happy Independence Day!!!";
}
else{
        echo "Nothing to show";
}
?>
```

Output:

```
Happy Independence Day!!!
```

SWITCH STATEMENT

The *switch* statement is similar to the series of if-else statements. The switch statement performs in various cases i.e. it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares it with the values of each case. If a case matches then the same case is executed.

To use the switch, we need to get familiar with two different keywords namely, break and default.

- **break:** The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
- **default:** The **default** statement contains the code that would execute if none of the cases match.

Syntax

```
switch (expression) {

  case label1:

    //code block

    break;

  case label2:

    //code block;

    break;

  case label3:

    //code block
    break;

  default:

    //code block

}
```

This is how it works:

- The *expression* is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break keyword breaks out of the switch block
- The default code block is executed if there is no match

Example

```
$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!";

}
```

WHILE LOOP

The **while** loop is the simple loop that executes nested statements repeatedly while the expression value is true. The expression is checked every time at the beginning of the loop, and if the expression evaluates to true then the loop is executed otherwise loop is terminated.

Syntax:

```
while (if the condition is true) {

    // Code is executed

}
```

```
<?php

     // Declare a number

    $num = 10;

     // While Loop

    while ($num < 20) {

        echo $num . "\n";

        $num += 2;

    }

?>
```

Output

```
10

12

14

16

18
```

FOR LOOP

The **for** loop is the most complex loop in PHP that is used when the user knows how many times the block needs to be executed. The **for** loop contains the initialization expression, test condition, and update expression (expression for increment or decrement).

Syntax:

```
for (initialization expression; test condition; update expression) {

   // Code to be executed

}
```

Loop Parameters:

- **Initialization Expression:** In this expression, we have to initialize the loop counter to some value. For example: $num = 1;
- **Test Condition:** In this expression, we have to test the condition. If the condition evaluates to “true” then it will execute the body of the loop and go to the update expression otherwise it will exit from the for loop. For example: $num <= 10;
- **Update Expression:** After executing the loop body, this expression increments/decrements the loop variable by some value. For example: $num += 2;

Example 1: The following code shows a simple example using **for** loop.

```
<?php

      // for Loop to display numbers

    for( $num = 0; $num < 20; $num += 5) {

        echo $num . "\n";

    }

  ?>
```

Output

```
0
5
10
15
```

UNIT – 3

PHP | Functions

A function is a block of code written in a program to perform some specific task. We can relate functions in programs to employees in a office in real life for a better understanding of how functions work. Suppose the boss wants his employee to calculate the annual budget. So how will this process complete? The employee will take information about the statistics from the boss, performs calculations and calculate the budget and shows the result to his boss. Functions works in a similar manner. They take informations as parameter, executes a block of statements or perform operations on this parameters and returns the result.
PHP provides us with two major types of functions:

- **Built-in functions** : PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions. To use those we just need to call them as per our requirement like, var_dump, fopen(), print_r(), gettype() and so on.
- **User Defined Functions** : Apart from the built-in functions, PHP allows us to create our own customised functions called the user-defined functions.
  Using this we can create our own packages of code and use it wherever necessary by simply calling it.

CREATING AN ARRAY

An array can hold more than one value. In PHP, they're stored as value pairs that in other languages would be called a dictionary or a hashtable. Keys can be strings or integers.

Syntax

There are several methods of declaring an array in PHP. The array() function can be used, either with key-value pairs, or with values alone. Single brackets, [...] can also be used in place of the array() keyword. If any key value is omitted, the key will be found by incrementing the largest prior integer key. If a key is repeated, the new value will overwrite the prior key.

Example

```
$cars = array("Volvo", "BMW", "Toyota");
```

A comma after the last item is allowed:

Example

```
$cars = [
  "Volvo",

  "BMW",

  "Toyota",

];
```

You can declare an empty array first, and add items to it later:

Example

```
$cars = [];

$cars[0] = "Volvo";

$cars[1] = "BMW";

$cars[2] = "Toyota";
```

MODIFYING ARRAY ELEMENTS

To update an existing array item, you can refer to the index number for indexed arrays, and the key name for associative arrays.

Example

Change the second array item from "BMW" to "Ford":

```
$cars = array("Volvo", "BMW", "Toyota");

$cars[1] = "Ford";
```

OUTPUT

```
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(4) "Ford"
  [2]=>
  string(6) "Toyota"
}
```

PROCESSING ARRAYS WITH LOOPS IN PHP

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

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

Example

Print $i as long as $i is less than 6:

```
$i = 1;

while ($i < 6) {

  echo $i;

  $i++;

}
```

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

Example

Print $i as long as $i is less than 6:

```
$i = 1;



do {

  echo $i;

  $i++;

} while ($i < 6);
```

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

Syntax

```
for (expression1, expression2, expression3) {

  // code block

}
```

This is how it works:

- *expression1* is evaluated once
- *expression2* is evaluated before each iteration
- *expression3* is evaluated after each iteration

Example

Print the numbers from 0 to 10:

```
for ($x = 0; $x <= 10; $x++) {

  echo "The number is: $x <br>";

}
```

The most common use of the foreach loop, is to loop through the items of an array. For every loop iteration, the value of the current array element is assigned to the variable $x. The iteration continues until it reaches the last array element.

Example:

```
<!DOCTYPE html>

<html>

<body>

<?php

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

foreach ($colors as $x) {

  echo "$x <br>";
}

?>

</body>

</html>
```

Output:

```
red
green
blue
yellow
```

GROUPING FORM SELECTIONS WITH ARRAYS

Groups an array into arrays by a given $key, or set of keys, shared between all array members. Array variables allow you to group related PHP data values together in a list using a single variable name. You can then either reference the values as a whole by referencing the variable name or handle each data value individually within the array by referencing its place in the list.

Parameters

- $array — The array to have grouping performed on.
- $key — The key to group or split by. Can be a *string*, an *integer*, a *float*, or a *callback*. If the key is *NULL*, the iterated element is skipped. If the key is a callback, it must return a valid key from the array.

  string|int callback (mixed $item)

- ... — Additional keys for grouping the next set of sub-arrays.

Return Values

Returns a multidimensional array, with each dimension containing elements grouped by the passed key(s).

Errors/Exceptions

If $key is not one of the accepted types E_USER_ERROR will be thrown and NULL returned.

Examples

Example #1 array_group_by() example

```
$records = [
	[
		"state"  => "IN",
		"city"   => "Indianapolis",
		"object" => "School bus"
	],
	[
		"state"  => "IN",
		"city"   => "Indianapolis",
		"object" => "Manhole"
	],
	[
		"state"  => "IN",
		"city"   => "Plainfield",
		"object" => "Basketball"
	],
	[
		"state"  => "CA",
		"city"   => "San Diego",
		"object" => "Light bulb"
	],
	[
		"state"  => "CA",
		"city"   => "Mountain View",
		"object" => "Space pen"
	]
];


$grouped = array_group_by( $records, "state", "city" );
```

The above example will output:

```
Array
(
	[IN] => Array
```

```
            (
                [Indianapolis] => Array
                    (
                        [0] => Array
                            (
                                [state] => IN
                                [city] => Indianapolis
                                [object] => School bus
                            )

                        [1] => Array
                            (
                                [state] => IN
                                [city] => Indianapolis
                                [object] => Manhole
                            )

                    )

                [Plainfield] => Array
                    (
                        [0] => Array
                            (
                                [state] => IN
                                [city] => Plainfield
                                [object] => Basketball
                    )

                )
            )

        [CA] => Array
            (
            [San Diego] => Array
                    (
                    [0] => Array
                        (
                                        [state] => CA
                                              [city] => San Diego
                                              [object] => Light bulb
                                        )
                            )

                      [Mountain View] => Array
                            (
                                  [0] => Array
                                        (
                                              [state] => CA
                                              [city] => Mountain View
                                              [object] => Space pen
                                  )
                            )
                )
)
```

USING ARRAY FUNCTIONS

PHP Arrays are a data structure that stores multiple elements of a similar type in a single variable. The arrays are helpful to create a list of elements of similar type. It can be accessed using their index number or key. The array functions are allowed to interact and manipulate the array elements in various ways. The PHP array functions are used for single and multi-dimensional arrays.

Installation: The array functions have not required any installation. These are part of PHP core concepts.

Function	**Description**
array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays

array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_assoc()	Compare arrays, and returns the differences (compare keys and values)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_fill()	Fills an array with values
array_keys()	Returns all the keys of an array
array_merge()	Merges one or more arrays into one array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_reverse()	Returns an array in the reverse order

array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values
count()	Returns the number of elements in an array
current()	Returns the current element in an array

UNIT – IV

PHP ADVANCED CONCEPTS -READING AND WRITING FILES

In PHP, reading from and writing to files is a common task in web development, especially for tasks such as reading configuration files, processing data files, or logging information. PHP provides built-in functions for handling file operations, making it straightforward to perform file read and write operations.

File Modes in PHP

Files can be opened in any of the following modes:

- **“w”** – Opens a file for writing only. If file does not exist then new file is created and if file already exists then file will be truncated (contents of file is erased).
- **“r”** – File is open for reading only.
- **“a”** – File is open for writing only. File pointer points to end of file. Existing data in file is preserved.
- **“w+”** – Opens file for reading and writing both. If file not exist then new file is created and if file already exists then contents of file is erased.
- **“r+”** – File is open for reading and writing both.
- **“a+”** – File is open for write/read. File pointer points to end of file. Existing data in file is preserved. If file is not there then new file is created.
- **“x”** – New file is created for write only.

Reading from a File:

- Use fopen() to open a file pointer.
- Use fgets() or fread( ) to read data from the file.
- Close the file pointer using fclose( ) when done.

Writing to a File:

- Use fopen() with mode 'w' or 'a' to open a file pointer for writing.
- Use fwrite() to write data to the file.
- Close the file pointer using fclose() when done.

Example (Reading from a File):

```
// Open a file for reading
$handle = fopen("file.txt", "r");
// Read data from the file
while (($line = fgets($handle)) !== false) {
    echo $line;
}
// Close the file handle
fclose($handle);
```

Example (Writing to a File):

```
// Open a file for writing (create if not exists, truncate if exists)
$handle = fopen("file.txt", "w");
// Write data to the file
fwrite($handle, "Hello, World!\n");
// Close the file handle
fclose($handle);
```

READING DATA FROM A FILE

PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character.

The available PHP file read functions are given below.

- fread()
- fgets()
- fgetc()

PHP Read File - fread()

The PHP fread() function is used to read data of the file. It requires two arguments: file resource and file size.

Syntax

string fread (resource $handle , int $length)
$handle represents file pointer that is created by fopen() function.

$length represents length of byte to be read.

Example

```
<?php
$filename = "c:\\file1.txt";
```

```
$fp = fopen($filename, "r");//open file in read mode
  $contents = fread($fp, filesize($filename));//read file
  echo "<pre>$contents</pre>";//printing data of file
fclose($fp);//close file
?>
```

Output

```
this is first line
this is another line
this is third line
```

PHP Read File - fgets()

The PHP fgets() function is used to read single line from the file.

Syntax

```
string fgets ( resource $handle [, int $length ] )
```

Example

```
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
echo fgets($fp);
fclose($fp);
?>
```

Output

```
this is first line
```

PHP Read File - fgetc()

The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax

```
string fgetc ( resource $handle )
```

Example

```
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
while(!feof($fp)) {
  echo fgetc($fp);
}
fclose($fp);
?>
```

Output

this is first line this is another line this is third line

UNIT – V

MANAGING SESSIONS AND USING SESSION VARIABLES

In general, session refers to a frame of communication between two medium. A PHP session is used to store data on a server rather than the computer of the user. Session identifiers or SID is a unique number which is used to identify every user in a session based environment. The SID is used to link the user with his information on the server like posts, emails etc.

Below are different steps involved in PHP sessions:

- **Starting a PHP Session**: The first step is to start up a session. After a session is started, session variables can be created to store information. The PHP **session_start()** function is used to begin a new session.It also creates a new session ID for the user.
  Below is the PHP code to start a new session:

```
<?php

  session_start();

  ?>
```

- **Storing Session Data**: Session data in key-value pairs using the **$_SESSION[]** superglobal array.The stored data can be accessed during lifetime of a session.
  Below is the PHP code to store a session with two session variables Rollnumber and Name:

```
<?php

  session_start();

  $_SESSION["Rollnumber"] = "11";

$_SESSION["Name"] = "Ajay";

  ?>
```

- **Accessing Session Data**: Data stored in sessions can be easily accessed by firstly calling **session_start()** and then by passing the corresponding key to the **$_SESSION** associative array.
  The PHP code to access a session data with two session variables Rollnumber and Name is shown below:

```
<?php

  session_start();

  echo 'The Name of the student is :' . $_SESSION["Name"] . '<br>';

echo 'The Roll number of the student is :' . $_SESSION["Rollnumber"] . '<br>';

  ?>
```

Output:

```
The Name of the student is :Ajay

The Roll number of the student is :11
```

- **Destroying Certain Session Data**: To delete only a certain session data,the unset feature can be used with the corresponding session variable in the **$_SESSION** associative array.
  The PHP code to unset only the “Rollnumber” session variable from the associative session array:

```
<?php

  session_start();

   if(isset($_SESSION["Name"])){

    unset($_SESSION["Rollnumber"]);

}

  ?>
```

- **Destroying Complete Session**: The **session_destroy()** function is used to completely destroy a session. The session_destroy() function does not require any argument.

```
<?php

  session_start();

session_destroy();

  ?>
```

1. The session IDs are randomly generated by the PHP engine .
2. The session data is stored on the server therefore it doesn’t have to be sent with every browser request.
3. The session_start() function needs to be called at the beginning of the page, before any output is generated by the script in the browser.

STORING DATA IN COOKIES

c**ookie** in PHP is a small file with a maximum size of 4KB that the web server stores on the client computer. They are typically used to keep track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.

Setting Cookie In PHP: To set a cookie in PHP, the **setcookie()** function is used. The setcookie() function needs to be called prior to any output generated by the script otherwise the cookie will not be set.

Syntax:

```
setcookie(name, value, expire, path, domain, security);
```

Parameters: The setcookie() function requires six arguments in general which are:

- **Name:** It is used to set the name of the cookie.
- **Value:** It is used to set the value of the cookie.
- **Expire:** It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
- **Path:** It is used to specify the path on the server for which the cookie will be available.
- **Domain:** It is used to specify the domain for which the cookie is available.
- **Security:** It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.

Below are some operations that can be performed on Cookies in PHP:

- **Creating Cookies**: Creating a cookie named Auction_Item and assigning the value Luxury Car to it. The cookie will expire after 2 days(2 days * 24 hours * 60 mins * 60 seconds).

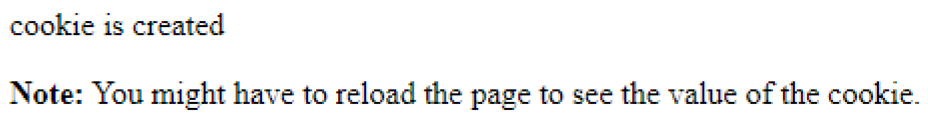
Example: This example describes the creation of the cookie in PHP.

- PHP

```
<!DOCTYPE html>

<?php

    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);

?>

<html>

<body>

    <?php

        echo "cookie is created."

    ?>
    <p>

        <strong>Note:</strong>

        You might have to reload the

        page to see the value of the cookie.

    </p>

  </body>

</html>
```

Note: Only the name argument in the setcookie() function is mandatory. To skip an argument, the argument can be replaced by an empty string(“”).

Output:



```
cookie is created

Note: You might have to reload the page to see the value of the cookie.
```

Cookie creation in PHP

SETTING COOKIES

Checking Whether a Cookie Is Set Or Not: It is always advisable to check whether a cookie is set or not before accessing its value. Therefore to check whether a cookie is set or not, the PHP isset() function is used. To check whether a cookie “Auction_Item” is set or not, the isset() function is executed as follows:

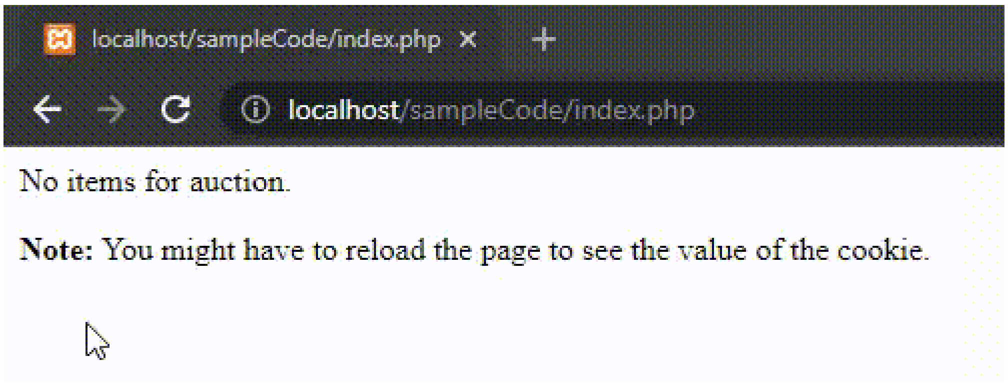
Example: This example describes checking whether the cookie is set or not.

- PHP

```
<!DOCTYPE html>

<?php

    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);

?>
```

```
<html>

<body>

   <?php

   if (isset($_COOKIE["Auction_Item"]))

   {         echo "Auction Item is a  " . $_COOKIE["Auction_Item"]; }

   else

   {         echo "No items for auction."; }

   ?>

   <p>

      <strong>Note:</strong>

      You might have to reload the page

      to see the value of the cookie.

   </p>

 </body>

</html>
```

Output:



localhost/sampleCode/index.php
localhost/sampleCode/index.php

```
No items for auction.

Note: You might have to reload the page to see the value of the cookie.
```

Checking for the cookie to be set