PHP Introduction:

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP, which originally stood for "Personal Home Page" but now recursively stands for "PHP: Hypertext Preprocessor," is a widely used server-side scripting language designed specifically for web development.
- PHP code is executed on the server, and the result (usually HTML) is sent to the client’s browser.
- PHP is free to download and use.
- To understand its role, it's crucial to differentiate between client-side and server-side operations in web development:
  - **Client-Side (Front-end):**
    This refers to what the user sees and interacts with directly in their web browser. Technologies like HTML (for structure), CSS (for styling), and JavaScript (for interactivity) run on the user's computer (the "client"). When you click a button or fill out a form, client-side scripts might handle basic validation or visual changes.
  - **Server-Side (Back-end):**
    This refers to the operations that happen on the web server before any content is sent to the user's browser. This is where PHP plays its vital role. When a user requests a web page that contains PHP code, the web server (e.g., Apache, Nginx) passes that request to a PHP interpreter. The PHP interpreter then executes the PHP code.

Aspect	Server-side Scripting	Client-side Scripting
Execution Location	Runs on the web server	Runs on the user's browser
Language Examples	PHP, Python, ASP.NET, Node.js, Java	HTML, CSS, JavaScript
Purpose	Generate dynamic content, handle database interaction, manage user sessions, and perform server operations	Control user interface, validate form data, and improve user experience without needing to contact the server
Output	Typically, HTML, JSON, or XML sent to the browser	Directly affects the display and behaviour of web pages in the browser
Access to Server Resources	Can access server files, databases, and perform complex tasks	Cannot access server files or databases directly

Visibility to Users	Hidden from users, users cannot see the server-side code	Visible to users, users can view the source code from the browser
Page Reload	Often requires page reload or server communication for changes	Can update the page without reloading (using JavaScript/ AJAX)
Security	More secure as code runs on the server	Less secure (code can be seen and modified by users)
Examples of Use	User authentication, data storage, content management systems	Form validation, interactive forms, animations, dropdowns

Role of PHP in server-side web development:

- **Dynamic Content Generation:**
  - Unlike static HTML pages that display the same content to every user, PHP allows for the creation of dynamic content. This means the content of a web page can change based on various factors, such as user input, data from a database, or the time of day.
  - Examples: Displaying a user's personalized dashboard, showing product listings from an e-commerce catalog, or generating search results based on a query.
- **Database Interaction:**
  - One of PHP's most powerful features is its seamless ability to connect and interact with various databases (like MySQL, PostgreSQL, SQLite, etc.).
  - It allows web applications to store, retrieve, update, and delete data, which is fundamental for almost all modern web applications (e.g., user accounts, blog posts, product inventories, order details).
- **Form Handling and Data Processing:**
  - When a user submits a form on a website (e.g., registration, contact form, login), PHP is commonly used to process that data.
  - It can validate the input, sanitize it (to prevent security vulnerabilities), store it in a database, send emails, or perform other necessary actions based on the submitted information.
- **Session Management and User Authentication:**
  - PHP enables the management of user sessions, allowing a website to "remember" a user's state across multiple page requests. This is crucial for features like user logins, shopping carts, and personalized experiences.
  - It handles user authentication (verifying identity) and authorization (determining what a user is allowed to do).
- **File System Operations:**
  - PHP can interact with the server's file system. This means it can read from,

write to, and manage files on the server.
  - Examples: Handling file uploads (like profile pictures), generating and managing log files, or creating dynamic reports.

- **Integration with Other Technologies:**
  - PHP can easily integrate with other technologies, including client-side scripts (HTML, CSS, JavaScript), external APIs (Application Programming Interfaces) for services like payment gateways or social media, and various server environments.
- **Security:**
  - By keeping sensitive operations and data on the server, PHP contributes to the security of web applications. Client-side code is visible to the user, but PHP code remains on the server, preventing direct access to sensitive logic or database credentials.

NOTE: In essence, PHP acts as the "brain" of many websites and web applications. It handles all the complex logic, data management, and communication with the server's resources, ultimately generating the HTML (and sometimes CSS/JavaScript) that the user's browser then displays.

Characteristics of PHP

Five important characteristics make PHP 's practical nature possible –

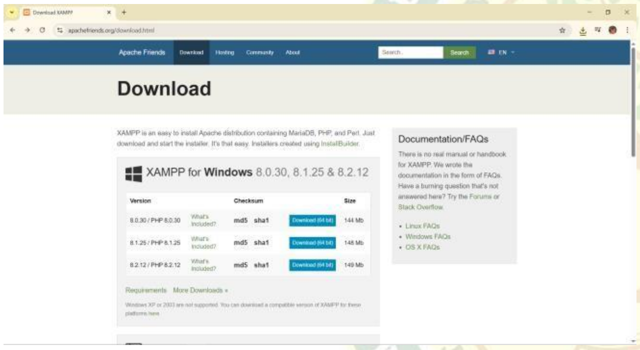
- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

History and evolution of PHP:

Year	Version	Key Features
1994	Creation of PHP Tools	Rasmus Lerdorf created some CGI scripts in C to track visitors
1995	PHP/FI (Personal Home Page / Forms Interpreter)	Released as PHP/FI 2.0, included HTML embedding and basic form handling
1997-1998	PHP 3	Full rewrite by Andi Gutmans and Zeev Suraski, introduced extensibility and improved performance, renamed to "PHP: Hypertext Preprocessor"

2000	PHP 4	Powered by the new "Zend Engine 1.0", improved speed and performance, better support for complex applications
2004	PHP 5	Introduction of Object-Oriented Programming (OOP), better XML support, PDO (PHP Data Objects) for database interaction, and improved error handling
2015	PHP 7	Major performance boost (up to 2x faster), reduced memory usage, introduced scalar type declarations, return type declarations, and error handling improvements
2020 onwards	PHP 8	Introduced JIT (Just-In-Time compilation) for faster execution, attributes (annotations), union types, match expressions, and other modern programming features

Installation and configuration using XAMPP/WAMP:


apachefriends.org/download.html

Download

XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy. Installers created using InstallBuilder.

XAMPP for Windows 8.0.30, 8.1.25 & 8.2.12

Version		Checksum		Size
8.0.30 / PHP 8.0.30	What's included?	md5 sha1	Download (64 bit)	144 Mb
8.1.25 / PHP 8.1.25	What's included?	md5 sha1	Download (64 bit)	148 Mb
8.2.12 / PHP 8.2.12	What's included?	md5 sha1	Download (64 bit)	149 Mb

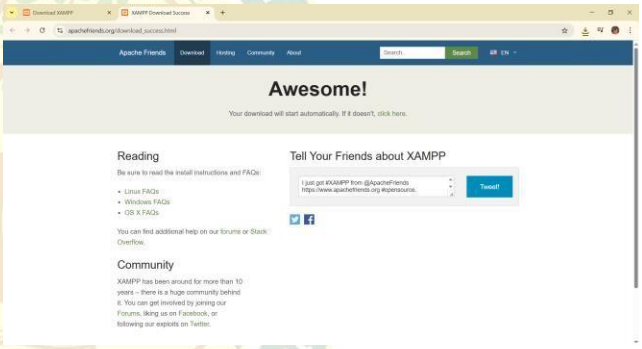
Requirements More Downloads »

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

Documentation/FAQs

There is no real manual or handbook for XAMPP. We wrote the documentation in the form of FAQs. Have a burning question that's not answered here? Try the Forums or Stack Overflow.

- Linux FAQs
- Windows FAQs
- OS X FAQs


apachefriends.org/download_success.html

Awesome!

Your download will start automatically. If it doesn't, click here.

Reading

Be sure to read the install instructions and FAQs:

- Linux FAQs
- Windows FAQs
- OS X FAQs

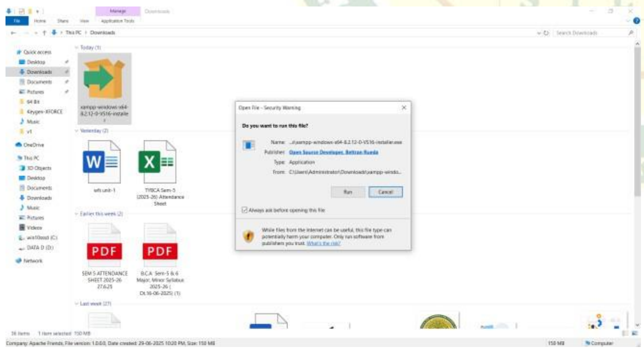
You can find additional help on our forums or Stack Overflow.

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, liking us on Facebook, or following our exploits on Twitter.

Tell Your Friends about XAMPP

I just got #XAMPP from @ApacheFriends https://www.apachefriends.org #opensource.


Open File - Security Warning

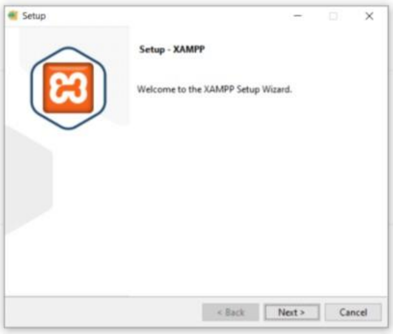
Do you want to run this file?

Name: ...\xampp-windows-x64-8.2.12-0-VS16-installer.exe
Publisher: Open Source Developer, Beltran Rueda
Type: Application
From: C:\Users\Administrator\Downloads\xampp-windo...

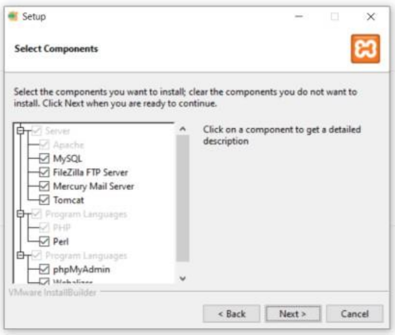
☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

Company: Apache Friends, File version: 1.0.0.0, Date created: 29-06-2025 10:20 PM, Size: 150 MB


Setup - XAMPP

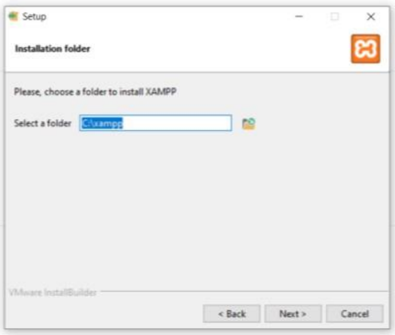
Welcome to the XAMPP Setup Wizard.


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.

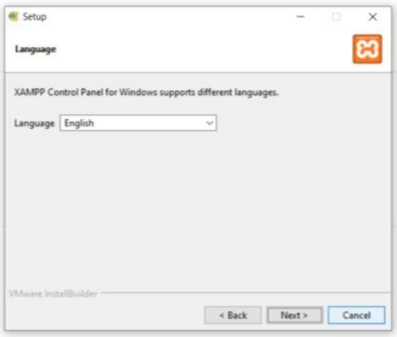
- ☑ Server
  - ☑ Apache
  - ☑ MySQL
  - ☑ FileZilla FTP Server
  - ☑ Mercury Mail Server
  - ☑ Tomcat
- ☑ Program Languages
  - ☑ PHP
  - ☑ Perl
- ☑ Program Languages
  - ☑ phpMyAdmin
  - ☑ Webalizer

Click on a component to get a detailed description


Installation folder

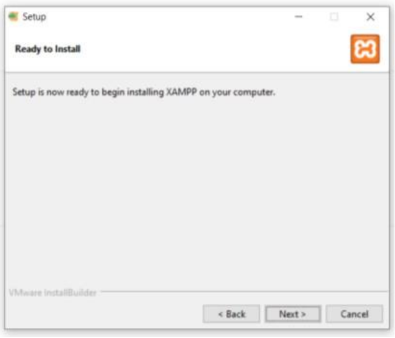
Please, choose a folder to install XAMPP

Select a folder C:\xampp


Language

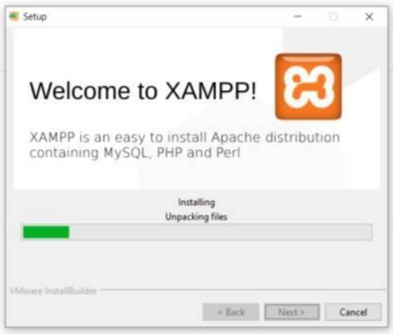
XAMPP Control Panel for Windows supports different languages.

Language English



Ready to Install

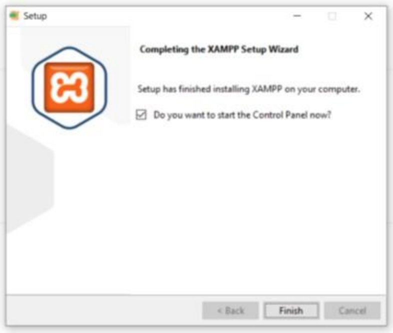
Setup is now ready to begin installing XAMPP on your computer.



Welcome to XAMPP!

XAMPP is an easy to install Apache distribution containing MySQL, PHP and Perl

Installing
Unpacking files



Completing the XAMPP Setup Wizard

Setup has finished installing XAMPP on your computer.

[✓] Do you want to start the Control Panel now?

Echo Keyword

echo is a language construct (not a function) used to output one or more strings. As echo is not a function you do not required to use parentheses with it. However, if you want to pass more than one parameter to echo(), using parentheses will generate a parse error.

Source Code	***Output***

`<html>`
`<head>`
`<title>Hello World</title>`
`</head>`
`<body>`
        `<?php echo "Hello, World!"; ?>`
`</body>`
`</html>`
	Hello, World!

echo and print statement

“echo” and “print” are more or less the same. They are both used to output data to the screen. “echo” has no return value while “print” has a return value of 1 so it can be used in expressions. “echo” can take multiple parameters (although such usage is rare) while “print” can take one argument. “echo” is marginally faster than “print”.

Echo	**Print**
1. echo does not return any value. 2. We can pass multiple strings separated by comma (,) in echo. <?php $name = “Alex”; #output → “Hello Alex!!!”	1. print always returns an integer value, which is 1. 2. Using print, we cannot pass multiple arguments. <?php $name = “Alex”; #this statement will give error

echo "Hello ", $name, "!!!"; ?>	print "Hello ", $name; ?>
3. echo is faster than print statement.	3. print is slower than echo statement.
4. Cannot be used in expression. <?php $x = 5; #this statement will give error ($x>0) ? echo('positive') : echo('negative'); ?>	4. Can be used in expression. <?php $x = 5; #output → "positive" ($x>0) ? print('positive') : print('negative'); ?>

Comments:

- Comments in any computer program (such as a PHP program) is a certain explanatory text that is ignored by the language compiler/interpreter.
- Its purpose is to help the user understand the logic used in the program algorithm.
- Although placing comments in the code is not essential, it is a highly recommended practice.
- The comments also serve as program documentation.
- Comments are also useful when the code needs to be debugged and modified.
- Single line comments are provided using # or //
  - Ex: //This is a comment OR #This is a comment
- The multiline style of commenting is the same as in C. One or more lines embedded inside the "/*" and "*/" symbols are treated as a comment.
  - Ex:
    ```
    /* This is a
    multi-line comment
    for better understanding */
    ```

Variables:

- 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) Note: Remember that PHP variable names are case-sensitive!

Variable Scope

PHP has three types of variable scopes:

- Local variable
- Global variable
- Static variable

Local Variable	Global Variable	Static Variable

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function.
 	
`<?PHP`
`function myTest() {`
`$x = 5;`
`// local scope`
 	
`echo "<p>Variable x inside function is: $x</p>";`
 	
`}`
`myTest();`
 	
`// using x outside the function will generate an error`
 	
`echo "<p>Variable x outside function is: $x</p>";`
`?>`
	A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.
	
	`<?PHP`
	`$name = "Sanaya Sharma";`
	`//Global Variable`
	
	`function global_var()`
	`{`
	`global $name;`
	`echo "Variable inside the function: ". $name;`
	`echo "</br>";`
	`}`
	`global_var();`
	`echo "Variable outside the function: ". $name;`
	`?>`
		It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as static variable. Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope.

	Another way to use the global variable inside the function is predefined $GLOBALS array.
	
	<?PHP
	$num1 = 5;
	//global variable
	
	$num2 = 13;
	
	//global variable
	
	function global_var()
	{
	$sum = $GLOBALS['num1'] + $GLOBALS['num2'];
	
	echo "Sum of global variables is: " .$sum;
	
	}
	
	global_var();
	
	?>
		<?PHP
		function static_var()
		{
		static $num1 = 3;
		//static variable
		$num2 = 6;
		//Non-static variable
		//increment in non-static variable
		$num1++;
		
		//increment in static variable
		$num2++;
		
		echo "Static: " .$num1 ."</br>"; echo "Non-static: " .$num2 ."</br>";
		}
		
		//first function call
		static_var();
		
		//second function call static_var();
		?>
		
		OUTPUT
		Output:
		Static: 4
		Non-static: 7
		Static: 5
		Non-static: 7

PHP $ and $$ Variables:

The $var (single dollar) is a $ variable is a normal PHP variable that contains a value. A variable can be allocated a wide range of values, including numbers, texts and arrays.

The $$var (double dollar) is a dynamic variable that takes the value of a normal variable and treats that as the name of the variable.

Source Code:	**Output**:

`<?php`
`$x="Hello";`
`$$x=200;`
 	
`echo $x . "</br>";`
`echo $$x."</br>;`
`echo $Hello;`
`?>`
	Hello
	200
	200

Constants

- By default, a PHP constant is case-sensitive.
- By convention, constant identifiers are always uppercase.
- A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscore.
- There is no need to write a dollar sign ($) before a constant, however one has to use a dollar sign before a variable.
- Constants are global, so they can be accessed from anywhere in the script.

PHP constants are name or identifier that can't be changed during the execution of the script except.

1. Using define() function
2. Using const keyword

Syntax:

define (name, value)

Example:

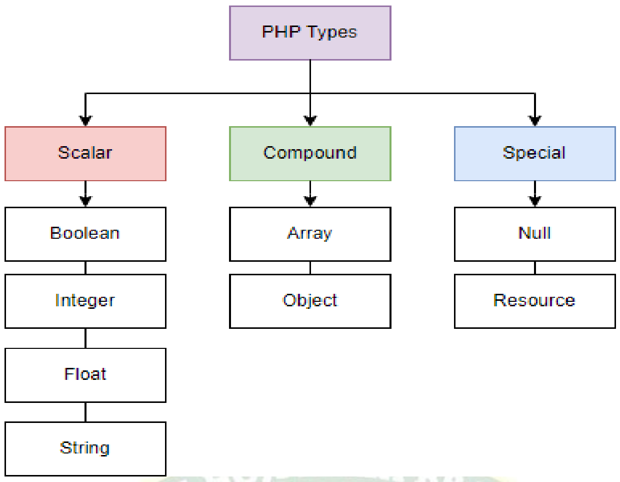
```
<?PHP
define("MSG1","Hello PHP ");
echo MSG1;
const MSG2="Hello const by PHP ";
echo MSG2;
?>
```

Data types

A type specifies the amount of memory that allocates to a value associated with it.

- Scalar Datatype: It holds single value only

- Compound Datatype: It holds multiple values
- Special Datatype: Used for specific use



Datatype	Description
Boolean	Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**
Integer	Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points. The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^{31} to +2^{31}
Float	A floating-point number is a number with a decimal point.
String	A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters. String values must be enclosed either within single quotes or in double quotes. But both are treated differently.

Source Code	Output

`<?php`
`$txt1 = "Learn PHP";`
`$txt2 = "Teach PHP";`
`echo "Hello $txt1";echo '<br/>';`
`echo 'Hello $txt2';`
`?>`
	Hello Learn PHP
	Hello $txt2

Array	An array is a compound data type. It can store multiple values of same data type in a single variable. $scores = [1, 2, 3];
Objects	Objects are the instances of user-defined classes that can store both values and functions.
Resource	Resources are not the exact data type in PHP . Basically, these are used to store some function calls or references to external PHP resources. For example - a database call.
NULL	Null is a special data type that has only one value: NULL

var_dump(variable name)

This function is used to dump information about a variable.

INPUT:

```
<?php
$a=123;
$b="Hello";
$c='123';
$d="123";
?>
```

OUTPUT:

```
int(123) string(5) "Hello" string(3)
"123" string(3) "123"
```

/* As mentioned in above source code first display datatype and length for string variable and datatype and value in Integer data type */

INPUT:

```
<?php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
?>
```

OUTPUT:

```
 array(3) {
   [0]=> int(1)
   [1]=> int(2)
   [2]=>array(3) {
     [0]=> string(1) "a"
     [1]=> string(1) "b"
     [2]=> string(1) "c"
   }
}
```

Type Casting

Type casting in PHP is a way to convert a value from one data type to another. It's often used when you need to perform an operation that requires a specific data type, or when you want to ensure that a variable holds a certain type of data.
Here are the different types of casts available in PHP:

- `(int)`, `(integer)`: Casts to integer

- `(bool)`, `(boolean)`: Casts to boolean
- `(float)`, `(double)`, `(real)`: Casts to float
- `(string)`: Casts to string
- `(array)`: Casts to array
- `(object)`: Casts to object
- `(unset)`: Casts to NULL **(available in PHP 7.2.0 and later, deprecated in PHP 8.0.0, removed in PHP 9.0.0)**

How it works:

You place the desired type in parentheses before the variable you want to cast.

Example:

```
<?php
$number = "123";                          // This is a string
$integer_number = (int) $number;          // Casts to integer

echo gettype($number);                    // Output: string
echo gettype($integer_number);            // Output: integer

$float_number = 10.5;                     // This is a decimal value
$int_from_float = (int) $float_number;    // Cast to integer, removes the decimal part

echo $int_from_float;                     // Output: 10

$string_boolean = "true";                 // This is a Boolean value
$boolean_value = (bool) $string_boolean;  // Casts to boolean

var_dump($boolean_value);                 // Output: bool(true)

$array_from_string = (array) "hello";     // This is an array
print_r($array_from_string);              // Output: Array ( [0] => hello )
?>
```

Important considerations:

- **Loss of data:** When casting from a more precise type (like float) to a less precise type (like integer), you might lose data (e.g., the decimal part of a float).

- **Behavior with different types:** The behavior of type casting can vary depending on the original data type and the target data type. For instance, converting a non-numeric string to an integer will result in `0`.
- **Temporary conversion:** Type casting creates a *copy* of the variable in the new type; it does not change the original variable's type. If you want to permanently change the type of a variable, you'd reassign it: `$variable = (new_type) $variable;`

Operators

PHP divides the operators in the following groups:

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

Operator	Name	Example	Explanation
+	Addition	$a + $b	Sum of operands
-	Subtraction	$a - $b	Difference of operands
*	Multiplication	$a * $b	Product of operands
/	Division	$a / $b	Quotient of operands
%	Modulus	$a % $b	Remainder of operands
**	Exponentiation	$a ** $b	$a raised to the power $b

Assignment operators: The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$a = $b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$a +=$b	Addition same as $a = $a + $b
-=	Subtract then Assign	$a =$b	Subtraction same as $a = $a - $b
*=	Multiply then Assign	$a *= $b	Multiplication same as $a = $a * $b
/=	Divide then Assign (quotient)	$a/=$b	Find quotient same as $a = $a / $b
%=	Divide then Assign (remainder)	$a%=$b	Find remainder same as $a = $a % $b

Conditional assignment operators: The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	$a & $b	Bits that are 1 in both $a and $b are set to 1, otherwise 0.
|	Or (Inclusive or)	$a | $b	Bits that are 1 in either $a or $b are set to 1
^	Xor (Exclusive or)	$a ^ $b	Bits that are 1 in either $a or $b are set to 0.
~	Not	~$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	$a << $b	Left shift the bits of operand $a $b steps
>>	Shift right	$a >> $b	Right shift the bits of $a operand by $b number of places

Comparison Operators: Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

Incrementing/Decrementing Operators: The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++$a	Increment the value of $a by one, then return $a
		$a++	Return $a, then increment the value of $a by one

--	decrement	--$a	Decrement the value of $a by one, then return $a
		$a--	Return $a, then decrement the value of $a by one

Logical Operators: The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
And	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
Xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

String Operators: The string operators are used to perform the operation on strings. There are two string operators in PHP , which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Array Operators: The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non-Identity	$a !== $b	Return TRUE if $a is not identical to $b

Conditional Statements

- To write code that perform different actions based on the results of a logical or comparative test condition at run time.
  - The **if** statement
  - The **if...else** statement
  - The if...elseif...else statement
  - The **switch... case** statement

The if Statement:

- The if statement is used to execute a block of code only if the specified condition evaluates to true.

Syntax:

```
if(condition) {
        // Code to be executed
}
```

Example:

```
<?PHP
$d = date("D");
if($d == "Fri") {
        echo "Have a nice weekend!";
}
?>
```

Output: "Have a nice weekend!" if the current day is Friday:

The if...else Statement

- You can enhance the decision-making process by providing an alternative choice through adding an else statement to the if statement.
- The if...else statement allows you to execute one block of code if the specified condition is evaluating to true and another block of code if it is evaluating to false.

Syntax:

```
if(condition) {
        // Code to be executed if condition is true
}
else {
        // Code to be executed if condition is false
}
```

Example:

```
<?PHP
$d = date("D");
if ($d == "Fri") {
        echo "Have a nice weekend!";
}
else {
        echo "Have a nice day!";
}
?>
```

Output: "Have a nice weekend!"
if the current day is Friday, otherwise it will
Output: "Have a nice day!"

The if...elseif...else Statement

- The if...elseif...else a special statement that is used to combine multiple if...else statements.

Syntax:

```
if(condition1) {
        // Code to be executed if condition1 is true
}
elseif (condition2) {
        // Code to be executed if the condition1 is false and condition2 is true
}
else
{
        // Code to be executed if both condition1 and condition2 are false
}
```

Example:

```
<?PHP
$d = date("D");
if ($d == "Fri") {
        echo "Have a nice weekend!";
}
elseif ($d == "Sun") {
        echo "Have a nice Sunday!";
}
else {
        echo "Have a nice day!";
}
?>
```

Output: "Have a nice weekend!"
if the current day is Friday, and

Output: "Have a nice Sunday!"
if the current day is Sunday, otherwise it will

Output: "Have a nice day!"

The switch-case statement:

- It is an alternative to the if-elseif-else statement, which does almost the same thing. The switch-case statement tests a variable against a series of values until it finds a match, and then executes the block of code corresponding to that match.

Syntax:

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

Example: Consider the following example, which display a different message for each day.

```
<?PHP
$today = date("D");
```

```
switch($today) {
        case "Mon":
                echo "Today is Monday. Clean your house.";
                break;
        case "Tue":
                echo "Today is Tuesday. Buy some food.";
                break;
        case "Wed":
                echo "Today is Wednesday. Visit a doctor.";
                break;
        case "Thu":
                echo "Today is Thursday. Repair your car.";
                break;
        case "Fri":
                echo "Today is Friday. Party tonight.";
                break;
        case "Sat":
                echo "Today is Saturday. Its movie time.";
                break;
        case "Sun":
                echo "Today is Sunday. Do some rest.";
                break;
        default:
                echo "No information available for that day.";
                break;
}
?>
```

Arrays

- It is used to hold multiple values of similar type in a single variable. Advantage of PHP Array
- **Less Code**: We don't need to define multiple variables.
- **Easy to traverse**: By the help of single loop, we can traverse all the elements of an array.
- **Sorting**: We can sort the elements of array.

There are 3 types of array in PHP.

1. Indexed Array 2. Associative Array 3. Multidimensional Array

Indexed Array

- PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.
- **There are two ways to define indexed array:**

```
$season = array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
            OR
$season[0] = "summer";
$season[1] = "winter";
$season[2] = "spring";
$season[3] = "autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
```

Associative Array

- We can associate name with each array elements in PHP using symbol.
- There are two ways to define associative array:

```
$salary = array("Sonoo"=>"350000", "John"=>"450000", "Kartik"=>"200000");
echo "Sonoo salary: " . $salary["Sonoo"] . "<br/>";
echo "John salary: " . $salary["John"] . "<br/>";
echo "Kartik salary: " . $salary["Kartik"] . "<br/>";
            OR
$salary["Sonoo"] = "350000";
$salary["John"] = "450000";
$salary["Kartik"] = "200000";
echo "Sonoo salary: " . $salary["Sonoo"] . "<br/>";
echo "John salary: " . $salary["John"] . "<br/>";
echo "Kartik salary: " . $salary["Kartik"] . "<br/>";
```

Multidimensional array

- An array containing one or more arrays and values are accessed using multiple indices.
- In a multidimensional array, each element in the main array can also be an array.

- Each element in the sub-array can be an array, and so on.

```
$AmazonProducts = array(
          array(“BOOK", "Books", 50),
          array("DVDs", “Movies", 15),
          array(“CDs", “Music", 20)
);

$AmazonProducts = array (
    array (“Code” =>“BOOK", “Description” => "Books", “Price” => 50),
    array (“Code” => "DVDs",“Description” => “Movies",“Price” => 15),
    array(“Code” => “CDs", “Description” => “Music", “Price” => 20)
);


for ($row = 0; $row < 3; $row++)
{
       echo $AmazonProducts[$row][“Code”] . “ ” .
    $AmazonProducts[$row][“Description”] . “ ” .
    $AmazonProducts[$row][“Price”];
}
```

Purpose	Syntax	Example
To access specific value	echo $array_name[Index];	echo $number[2]; echo $numbers[0] . " and " . $numbers[2] ;

To access all values	foreach (array_name as variable_name)
	{
	echo $variable_name;
	}
		foreach ($numbers as $value)
		{
		echo "Value is $value <br />";
		}

Accessing value from multidimensional array		for ($row = 0; $row < 3; $row++)
		{
		for ($column = 0; $column < 3; $column++)

		{
		echo $AmazonProducts [$row] [$column] . "<br />";
		}
		}

Sorts the values in array [it replaces key with 0,1,2,3]	sort($array_name)	
Sort the value but key-value associations are preserved.	asort($array)	
Sorts array by its keys, rather than by its values. Key-value associations are preserved.	Ksort()	
rsort($array), arsort($array), krsort($array) – functions behave the same as sort,asort and ksort respectively, but sort in the reverse order.		
Split string value in array	$array_name=explode("Delimiter","String"); // It takes an argument of a string and a delimiter and returns an array consisting of substrings of the string.	$str="a:b:c:d"; $arr=explode(":",$str);
Convert array into string	$string=implode("Delimiter","Array");	$str = implode("^",$ar"); $str has the value: "a^b^c^d".

	//It takes an array and returns a string where the entries are appended together using a delimiter	
Current pointer value	current($array_name)	$snacks=array("chips","candy"); $current_pointer=current($snacks); echo $ current_pointer;

Display next and previous pointer value	next($array_name)
	prev($array_name)
		$snack=next($snacks);
		print("The 2nd snack is $snack");
		//display candy
		$snack=prev($snacks);
		print("The 1st snack is $snack");
		//display chips

Moves "current" to the last element in the array and then dereferences it.	end($array_name)	
Moves an array's "current" pointer to the first element in the array and then dereferences it.	reset($array_name)	

Used to delete element as per assigned array index.	unset($array_name[index]);	$a=array(1,2,3);
		unset($a[0]);
		print_r($a);

creates an array by using the elements from one "keys" array and one "values" array. Note: Both arrays must have equal number of elements!	array_combine($array1,$array2);	$fname=array("Peter","Ben","Joe"); $age=array("35","37","43"); $c=array_combine($fname,$age); print_r($c);

PHP Loops

- 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.

Syntax

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

Example

```
<?PHP
$x = 1;
while($x <= 5) {
```

```
echo "The number is: $x <br>";
$x++;
}
?>
```

do...while Loop

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

Syntax

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

The example below first sets a variable $x to 1 ($x = 1).
Then, the do while loop will write some output.
Then increment the variable $x with 1.
Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?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.

Syntax

```
for (init counter; test counter; increment counter)
```

```
{
      code to be executed for each iteration;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

The example below displays the numbers from 0 to 10:

Example

```
<?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.

Syntax

```
foreach ($array as $value)
{
        code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

```
<?PHP
        $colors = array("red", "green", "blue", "yellow");
        foreach ($colors as $value)
```

```
	{
		echo "$value <br>";
	}
?>
```

Break

- You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.
- The break statement can also be used to jump out of a loop.
- This example jumps out of the loop when x is equal to 4:

Example

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

Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.
This example skips the value of 4:

Example

```
<?PHP
for ($x = 0; $x < 10; $x++)
{
	if ($x == 4)
	{
		continue;
	}
```

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

include()

- One of the most useful tools is to insert another php script from a file into the current php script.
- The command **include("filename");** will import contents of a text file called filename and insert it at the include spot.
- The included text may be composed of XHTML, PHP or both.
- The **include()** function is mostly used when the file is not required and the application should continue to execute its process when the file is not found.
- The **include()** function will only produce a warning (E_WARNING) and the script will continue to execute.

Example:

File 1: menu.php

```
<a href=”default.php”>HOME</a>
<a href=”contact.php”>CONTACT US </a>
<a href=”staff.php”>StAFF</a>
```

File 2 :Student.php

```
<html>
<head></head>
<body>
<?php
        include(“menu.php”);
        //If menu.php is not found then also remaining echo statement in script will be
executed.
?>
</body>
</html>
```

require()

- Syntax and uses is as same as include() but the difference is that, if the file is not found the remaining script is also not executed.
- The require() function is mostly used when the file is mandatory for the application.
- The **require()** will produce a fatal error (E_COMPILE_ERROR) along with the warning and the script will stop its execution.

Functions

- Function is a block of statement that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- Function will be executed by a call to the function.

Create function:

- A user-defined function declaration starts with a keyword function

Syntax:

```
function functionName(parameters)
{
   function-body
}
```

Example:

```
function generateFooter()
{
   echo "Copyright 2010 W. Jason Gilmore";
}
```

Once defined, you can call this function like so:

```
<?php
   generateFooter();
?>
```

Passing Arguments by Value

```
function calcSalesTax($price, $tax)
{
   $total = $price + ($price * $tax);
   echo "Total cost: $total";
}
```

Passing Arguments by Reference

```
<?php
   $cost = 20.99;
   $tax = 0.0575;
   function calculateCost(&$cost, $tax)
   {
      // Modify the $cost variable
      $cost = $cost + ($cost * $tax);
      // Perform some random change to the $tax variable.
      $tax += 4;
   }
   calculateCost($cost, $tax);
   printf("Tax is %01.2f%% ", $tax*100);
   printf("Cost is: $%01.2f", $cost);
?>
```

Form Handling

- There are two ways the browser client can send information to the web server.
    1. The GET Method
    2. The POST Method
- Before the browser sends the information, it encodes it using a scheme called URL encoding.
- In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.
    - name1=value1&name2=value2&name3=value3

GET Method

- The GET method sends the encoded user information appended to the page request.
- The page and the encoded information are separated by the "?" character.
  - The GET method produces a long string that appears in your server logs, in the browser's Location: box.
  - The GET method is restricted to send up to 1024 characters only.
  - Never use GET method if you have password or other sensitive information to be sent to the server.
  - GET can't be used to send binary data, like images or word documents, to the server.
  - The data sent by GET method can be accessed using QUERY_STRING environment variable.
  - The PHP provides $_GET associative array to access all the sent information using GET method.

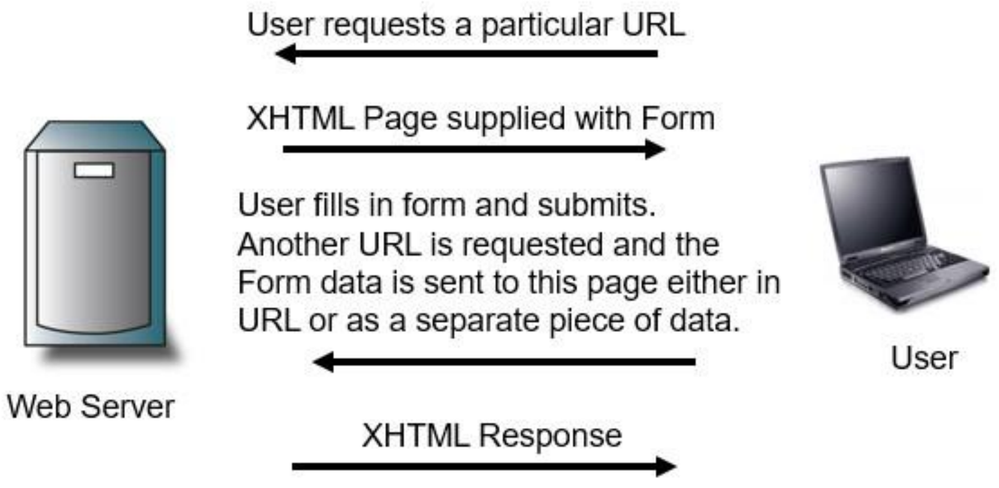
POST Method

- The POST method transfers information via HTTP headers.
  - The information is encoded as described in case of GET method and put into a header called QUERY_STRING.
  - The POST method does not have any restriction on data size to be sent.
  - The POST method can be used to send ASCII as well as binary data.
  - The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
  - The PHP provides $_POST associative array to access all the sent information using POST method.

Access Data

- Access submitted data in the relevant array for the submission type, using the input name as a key.

```
<form action="path/to/submit/page" method="get">
        <input type="text" name="email">
</form>
$email = $_GET['email'];
```


User requests a particular URL
XHTML Page supplied with Form
User fills in form and submits. Another URL is requested and the Form data is sent to this page either in URL or as a separate piece of data.
XHTML Response
Web Server
User

Work with XHTML Form

- The form is enclosed in form tags:

```
<form action="path/to/submit/page" method="get">
        <!-- form contents -->
</form>
```

  - **action="…"** is the page that the form should submit its data to.
  - **method="…"** is the method by which the form data is submitted. The option is either **get** or **post**. If the method is get the data is passed in the URL string, if the method is post it is passed as a separate file.

$_GET	**$_POST**
In GET method we cannot send large amount of data rather limited data of some number of characters is sent because the request parameter is appended into the URL.	In POST method large amount of data can be sent because the request parameter is appended into the body.
GET requests are only used to request data (not modify)	POST requests can be used to create and modify data.
GET request is comparatively less secure because the data is exposed in the URL bar.	POST request is comparatively more secure because the data is not exposed in the URL bar.

Request made through GET method are stored in Browser history.	Request made through POST method is not stored in Browser history.
GET method request can be saved as bookmark in browser.	POST method request cannot be saved as bookmark in browser.
In GET method only ASCII characters are allowed.	In POST method all types of data is allowed.
Request made through GET method are stored in cache memory of Browser.	Request made through POST method are not stored in cache memory of Browser.
Data passed through GET method can be easily stolen by attackers as the data is visible to everyone. GET requests should never be used when dealing with sensitive data	Data passed through POST method cannot be easily stolen by attackers as the URL Data is not displayed in the URL
In GET method, the Encoding type is application/x-www-form-urlencoded	In POSTmethod, the encoding type is *application/x-www-form-urlencoded* or *multipart/form-data*. Use multipart encoding for binary data

Example:
`<!DOCTYPE html>`
`<html>`
 	
`<body>`
`<form action="getmethod.php" method="GET">`
`Username:`
`<input type="text" name="username" /> <br>`
`City:`
`<input type="text" name="city" /> <br>`<br>`<input type="submit" />`
`</form>`
`</body>`
 	
`</html>`
	Example:
	`<!DOCTYPE html>`
	`<html>`
	
	`<body>`
	`<form action="postmethod.php" method="post">`
	`Username:`
	`<input type="text" name="username" /> <br>`<br>`Area of Study:`
	`<input type="text" name="area" /> <br>`
	<br>`<input type="submit" />`
	`</form>`
	`</body>`
	
	`</html>`

```
<!DOCTYPE html>
<html>

<body>
   Welcome
   <?php echo $_GET["username"]; ?>
</br>
   Your City is:
   <?php echo $_GET["city"]; ?>
</body>

</html>
```

```
<!DOCTYPE html>
<html>

<body>
   Welcome
   <?php echo $_POST["username"]; ?>
</br>
   Your Area of Study is:
   <?php echo $_POST["area"]; ?>
</body>

</html>
```

Basic Input Validation and Sanitization in PHP

When developing PHP applications, especially web forms, it's essential to **validate** and **sanitize** user input to prevent common security vulnerabilities such as **XSS (Cross-Site Scripting)**, **SQL Injection**, and **malformed data entry**.

1. Input Validation vs. Sanitization

- **Validation**: Ensures the data is of the correct type, format, and meets certain criteria (e.g., email format, number range).
- **Sanitization**: Cleans the data by removing unwanted characters or encoding it to prevent malicious code execution.

2. Common PHP Functions for Input Validation and Sanitization

Field	**Validation Rule**
Name	Required- Must only contain **letters** and **whitespace**
E-mail	Required- Must be a **valid email format** (must include @ and .)
Website	Optional- If provided, must be a **valid URL**
Gender	Required- Must select one option (e.g., Male/Female/Other)
Aadhar Card	Required- Must be a **12-digit number only** (exactly 12 digits, all numeric)

Type	Function	Description
Validation	filter_var()	Validates email, URLs, integers, etc.
Sanitization	htmlspecialchars()	Converts special HTML characters to entities
Sanitization	strip_tags()	Removes HTML and PHP tags
Validation	preg_match()	Validates strings using regular expressions
Both	filter_input()	Retrieves and filters external input (e.g., $_POST)

3. Example of Validation and Sanitization

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Sanitize name input
    $name = htmlspecialchars(strip_tags($_POST["name"]));

    // Validate email input
    $email = filter_var($_POST["email"], FILTER_VALIDATE_EMAIL);

    // Sanitize and validate integer age
    $age = filter_var($_POST["age"], FILTER_SANITIZE_NUMBER_INT);
    if (!filter_var($age, FILTER_VALIDATE_INT)) {
        echo "Invalid age.";
    }

    // Check if email is valid
    if ($email === false) {
        echo "Invalid email format.";
    } else {
        echo "Welcome, $name. Your email is $email and your age is $age.";
    }
}
?>
```