**INTRODUCTION**

1. **PHP Overview**
2. **PHP Comments**
3. **Variables in PHP**
4. **Variable Scope**

**Data Types & Operators**

1. **PHP DataTypes**
2. **PHP Operators**

**Conditional & Iterative Statements**

1. **PHP Conditional Statements**
2. **PHP Iterative Statements**

**PHP Functions**

1. **Function Basics**
2. **Function Arguments**

**PHP Arrays**

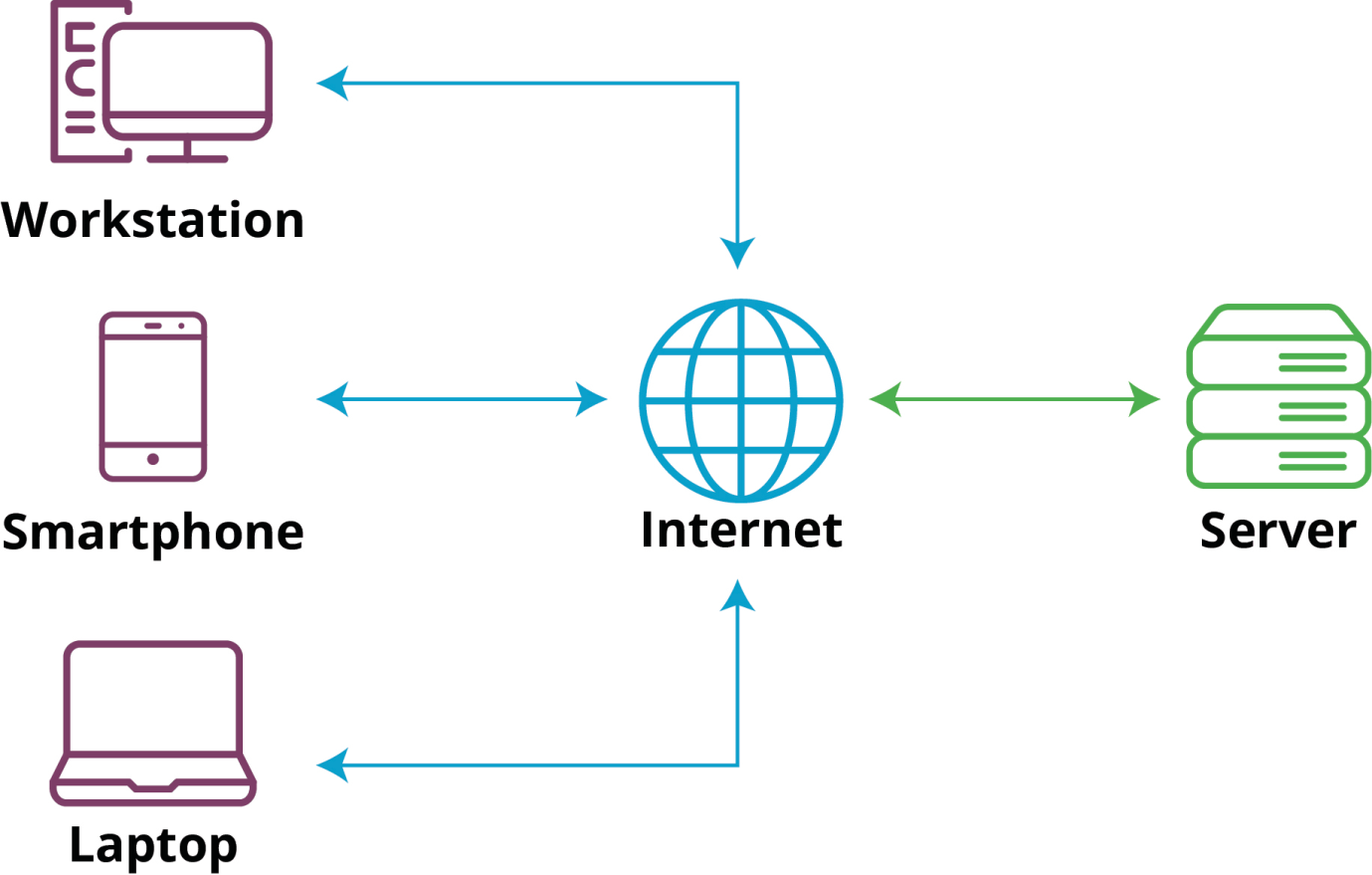
1. **Arrays**

**PHP Overview**

PHP (Hypertext Preprocessor) is a widely-used server-side scripting language designed for web development. Here's a brief overview of PHP:

PHP (Hypertext Preprocessor) was created by Rasmus Lerdorf.

The development began in 1994 Over time, Lerdorf continued to develop and improve PHP, adding more functionality and features. The first version that resembled what we now know as PHP was released in 1995



Client Server Architecture

What is server?

A server is a computer or software system that provides services, data, or resources to other computers, known as clients, over a network

1. **Purpose:**
   * PHP is mainly used for server-side scripting, enabling the creation of dynamic web pages and web applications.
   * It can be embedded in HTML code and executed on the server, producing HTML output sent to the client's browser.
2. **Key Features:**
   * **Open Source:** PHP is an open-source language, making it widely accessible and continuously evolving with community contributions.
   * **Cross-Platform:** PHP runs on various operating systems, including Windows, Linux, macOS, and can be easily integrated with web servers like Apache and Nginx.
   * **Database Integration:** PHP can interact with a variety of databases, including MySQL, PostgreSQL, SQLite, and others, facilitating dynamic content generation.

**Uses of PHP**

PHP can perform several system functions like opening files, CRUD operations on data stores, general-purpose scripting, etc. Besides system operations, there are also other uses like

1. Handling Forms: PHP can handle form operations. It can gather data, save data to a file and send data through emails.
2. Database Operations: PHP can also create, read, update and delete elements in your database.
3. Encryption: It can perform advanced encryption and encrypt data for you.
4. Dynamic Page Content: It can generate dynamic page content.

**Basic Syntax PHP**

A PHP script can be written anywhere inside the HTML document. A PHP script starts with <?php tag and ends with ?>. We can write our logic inside this tag and it will be executed accordingly.

**<?php**

// PHP code goes here

**?>**

**Displaying output in php**

In php,Output is displayed on the browser using echo as follows:

**<?php**

echo "hello";

**?>**

**Hello World**

A basic PHP Hello World program looks something like this. We will use a built-in PHP contruct or keyword “echo” to output the text “Hello World!” on our webpage.

<!DOCTYPE html>

<html>

<body>

<h1>My first PHP page</h1>

**<?php**

echo "Hello World!";

**?>**

</body>

</html>

In PHP, keywords are reserved words that have special meanings and cannot be used as identifiers (such as variable names, function names, class names, etc.). Here is a list of some PHP keywords:

1. **Basic Language Constructs:**
   * **echo**
   * **print**
   * **unset**
   * **return**
   * **include**
   * **include\_once**
   * **require**
   * **require\_once**
   * **global**
   * **if**
   * **else**
   * **elseif** (alternative to **else if**)
   * **switch**
   * **case**
   * **default**
   * **while**
   * **do**
   * **for**
   * **foreach**
   * **break**
   * **continue**
   * **declare**
   * **function**
   * **class**
   * **interface**
   * **trait**
   * **new**
   * **instanceof**
   * **extends**
   * **implements**
   * **namespace**
   * **use**
   * **const**
   * **final**
   * **abstract**
   * **static**

**PHP Comments**

A comment is a part of the coding file that the programmer does not want to execute, rather the programmer uses it to either explain a block of code or to avoid the execution of a specific part of code while testing.

**PHP supports several ways of commenting:**

**Single Line Comments**

**<?php**

// This is a single-line comment

# This is also a single-line comment

**?>**

**Multiple-Line Comments**

$x = 5 /\* + 15 \*/ + 5;

echo $x;

The  + 15 part will be ignored in the calculation:

**<?php**

/\*

This is a

multiple line

Comment.

\*/

**?>**

**Variables in PHP**

**Variables are containers that can store information which can be manipulated or referenced later by the programmer within the code.**

In PHP, we don’t need to declare the variable type explicitly. The type of variable is determined by the value it stores. There are some important things to know about variables in PHP.

* All variables should be denoted with a Dollar Sign ($)
* Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
* Variable names can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ ).
* Variables must start with a letter or the underscore “\_” character.
* Variables are case sensitive
* Variable names cannot start with a number.

**For Example:**

**<?php**

$txt = "Hello world!"; # Type String

$x = 5; # Type int

$y = 10.5; # Type Float

**?>**

**Note:** When you assign a text value to a variable, put quotes around the value.

**Note:** Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

$txt = "programming";

echo "I love $txt!";

The following example will output the sum of two variables:

### Example

$x = 5;

$y = 4;

echo $x + $y;

## PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.

## Variable Types

PHP has no command for declaring a variable, and the data type depends on the value of the variable.

### Example

$x = 5; // $x is an integer

$y = "John"; // $y is a string

echo $x;

echo $y;

PHP supports the following data types:

* String "some text"
* Integer 8
* Float (floating point numbers - also called double) 8.5
* Boolean 0 or 1
* Array [12,23,34]
* Object
* NULL
* Resource

## Get the Type

To get the data type of a variable, use the var\_dump() function.

### Example

The var\_dump() function returns the data type and the value:

$x = 5;

var\_dump($x);

See what var\_dump() returns for other data types:

var\_dump(5);

var\_dump("John");

var\_dump(3.14);

var\_dump(true);

var\_dump([2, 3, 56]);

var\_dump(NULL);

## Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

### Example

$x = "John";

echo $x;

## Assign Multiple Values

You can assign the same value to multiple variables in one line:

### Example

All three variables get the value "Fruit":

$x = $y = $z = "Fruit";

**Variable Scope**

The scope of the variable is the area within which the variable has been created. Based on this a variable can either have a local scope or a global scope or a static scope in PHP.

**Global Variable:**

A variable which was created in the main body of the code and that can be accessed anywhere in the program is called Global Variable. Global variables can be directly accessed or used in or outside of a function with GLOBAL keyword before variable. However, we can also call them without the global keyword.

**For Example:**

**<?php**

$name = "hello Bhai"; //Global Variable

function global\_var()

{

global $name;

echo "Variable inside the function: ". $name;

echo "</br>";

}

global\_var();

echo "Variable outside the function: ". $name;

**?>**

Output:

Variable inside the function: hello Bhai

Variable outside the function: hello Bhai

**Local Variable:**

A local variable is created within a function and can be only used inside the function. This means that these variables cannot be accessed outside the function, as they have local scope.

**For Example:**

**<?php**

function mytest()

{

$capital = "Delhi";

echo "Capital of India is: " .$capital;

}

mytest(); //Calling the function

//using $capital outside the function will generate an error

echo $capital;

**?>**

Output:

Capital of India is: Delhi Notice: Undefined variable: capital in D:\xampp\htdocs\program\var.php on line 28

**Static Variable:**

PHP has a feature that deletes the variable once it has finished execution and frees the memory. When we need a local variable which can store its value even after the execution, we use the static keyword before it and the variable is called static variable.

These variables only exist in a local function and do not get deleted after the execution has been completed.

**For Example:**

**<?php**

function static\_var()

{

static $num1 = 3; //static variable

$num2 = 6; //Non-static variable

//increment in non-static variable which will increment its value to 7

$num1++; // $num1 = $num1 + 1;

//increment in static variable which will increment its value to 4 after first execution and 5 after second execution

$num2++;

echo "Static: " .$num1 ."</br>";

echo "Non-static: " .$num2 ."</br>";

}

//first function call

static\_var();

//second function call

static\_var();

**?>**

Output:

Static: 4

Non-static: 7

Static: 5

Non-static: 7

PHP echo and print Statements

With PHP, there are two basic ways to get output: echo and print.

In this tutorial we use echo or print in almost every example. So, this chapter contains a little more info about those two output statements.

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

## The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

**Display Text**

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

  echo ("some value second value "),("$a");

### echo "<h2>PHP is Fun!</h2>";

echo "Hello world!<br>";

echo "I'm about to learn PHP!<br>";

echo "This ", "string ", "was ", "made ", "with multiple parameters.";

**Display Variables**

The following example shows how to output text and variables with the echo statement:

### Example

$txt1 = "Learn PHP";

$txt2 = "shubham";

$x = 5;

$y = 4;

echo "<h2>" . $txt1 . "</h2>";

echo "Study PHP with " . $txt2 . "<br>";

echo $x + $y;

## The PHP print Statement

The print statement can be used with or without parentheses: print or print().

**Display Text**

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

### Example

print "<h2>PHP is Fun!</h2>";

print "Hello world!<br>";

print "I'm about to learn PHP!";

**Display Variables**

The following example shows how to output text and variables with the print statement:

### Example

$txt1 = "Learn PHP";

$txt2 = " Schools.com";

$x = 5;

$y = 4;

print "<h2>" . $txt1 . "</h2>";

print "Study PHP at " . $txt2 . "<br>";

print $x + $y;

## PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource

## Getting the Data Type

You can get the data type of ant object by using the var\_dump() function.

### Example

The var\_dump() function returns the data type and the value:

$x = 5;

var\_dump($x);

**PHP String**

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

### Example

$x = "Hello world!";

$y = 'Hello world!';

var\_dump($x);

echo "<br>";

var\_dump($y);

## PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

* An integer must have at least one digit
* An integer must not have a decimal point
* An integer can be either positive or negative
* Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var\_dump() function returns the data type and value:

### Example

$x = 5985;

var\_dump($x);

## PHP Float

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

In the following example $x is a float. The PHP var\_dump() function returns the data type and value:

### Example

$x = 10.365;

var\_dump($x);

## PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

### Example

$x = true;

var\_dump($x);

Booleans are often used in conditional testing.

You will learn more about conditional testing in the [PHP If...Else chapter](https://www.w3schools.com/php/php_if_else.asp).

## PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var\_dump() function returns the data type and value:

### Example

$cars = array("Volvo","BMW","Toyota",65,565.76);

var\_dump($cars);

You will learn a lot more about arrays in later chapters of this tutorial.

## PHP Object

Classes and objects are the two main aspects of object-oriented programming.

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

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car that can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a \_\_construct() function, PHP will automatically call this function when you create an object from a class.

### Example

class Car {

public $color;

public $model;

public function \_\_construct($color, $model) {

$this->color = $color;

$this->model = $model;

}

public function message() {

return "My car is a " . $this->color . " " . $this->model . "!";

}

}

$myCar = new Car("red", "Volvo");

var\_dump($myCar);

In PHP, the **->** (arrow) is an operator used to access properties and methods of an object. It is used to reference a member of an object

Do not worry if you do not understand the PHP Object syntax, you will learn more about that in the [PHP Classes/Objects chapter](https://www.w3schools.com/php/php_oop_classes_objects.asp).

## PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

**Tip:** If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

### Example

$x="hello world!";

$x = null;

var\_dump($x);

## Change Data Type

If you assign an integer value to a variable, the type will automatically be an integer.

If you assign a string to the same variable, the type will change to a string:

### Example

$x = 5;

var\_dump($x);

$x = "Hello";

var\_dump($x);

If you want to change the data type of an existing variable, but not by changing the value, you can use casting.

Casting allows you to change data type on variables:

### Example

$x = 5;

$x = (string) $x;

var\_dump($x);

You will learn more about casting in the [PHP Casting Chapter](https://www.w3schools.com/php/php_casting.asp).

## PHP Resource

The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP.

A common example of using the resource data type is a database call.

We will not talk about the resource type here, since it is an advanced topic.

PHP Strings and functions:-

Strings in PHP are surrounded by either double quotation marks, or single quotation marks.

### Example

echo "Hello";

echo 'Hello';

**Note** There is a big different between double quotes and single quotes in PHP.

Double quotes process special characters, single quotes does not.

## Double or Single Quotes?

You can use double or single quotes, but you should be aware of the differences between the two.

Double quoted strings perform action on special characters.

E.g. when there is a variable in the string, it returns the value of the variable:

### Example

Double quoted string literals perform operations for special characters:

$x = "John";

echo "Hello $x";

Single quoted strings does not perform such actions, it returns the string like it was written, with the variable name:

Example

Single quoted string literals returns the string as it is:

$x = "John";

echo 'Hello $x';

Some String functions

1. strlen() returns the length of a string
2. str\_word\_count() function counts the number of words in a string
3. strpos("","") function searches for a specific text within a string

# Modify Strings:-

PHP has a set of built-in functions that you can use to modify strings.

**Upper Case**

### Example

The strtoupper() function returns the string in upper case:

$x = "Hello World!";

echo strtoupper($x);

## Lower Case

### Example

The strtolower() function returns the string in lower case:

$x = "Hello World!";

echo strtolower($x);

## Replace String

The PHP str\_replace() function replaces some characters with some other characters in a string.

### Example

Replace the text "World" with "Dolly":

$x = "Hello World!";

echo str\_replace("World", "Dolly", $x);

## Reverse a String

The PHP strrev() function reverses a string.

### Example

Reverse the string "Hello World!":

$x = "Hello World!";

echo strrev($x);

## Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

### Example

The trim() removes any whitespace from the beginning or the end:

$x = " Hello World! ";

echo trim($x);

Learn more in our [trim() Function Reference](https://www.w3schools.com/php/func_string_trim.asp).

## Convert String into Array

The PHP explode() function splits a string into an array.

The first parameter of the explode() function represents the "separator". The "separator" specifies where to split the string.

**Note:** The separator is required.

### Example

Split the string into an array. Use the space character as separator:

$x = "Hello World!");

$y = explode(" ", $x);

//Use the print\_r() function to display the result:

print\_r($y);

/\*

Result:

Array ( [0] => Hello [1] => World! )

\*/

# PHP - Concatenate String:-

## String Concatenation

To concatenate, or combine, two strings you can use the . operator:

### Example

$x = "Hello";

$y = "World";

$z = $x . $y;

echo $z;

The result of the example above is HelloWorld, without a space between the two words.

You can add a space character like this:

### Example

$x = "Hello";

$y = "World";

$z = $x . " " . $y;

echo $z;

An easier and better way is by using the power of double quotes.

By surrounding the two variables in double quotes with a white space between them, the white space will also be present in the result:

### Example

$x = "Hello";

$y = "World";

$z = "$x $y";

echo $z;

## String Slicing:-

You can return a range of characters by using the substr() function.

Specify the start index and the number of characters you want to return.

### Example

Start the slice at index 6 and end the slice 5 positions later:

$x = "Hello World!";

echo substr($x, 6, 5);

**Note** The first character has index 0.

## Slice to the End

By leaving out the length parameter, the range will go to the end:

### Example

Start the slice at index 6 and go all the way to the end:

$x = "Hello World!";

echo substr($x, 6);

## Slice From the End

Use negative indexes to start the slice from the end of the string:

### Example

Get the 3 characters, starting from the "o" in world (index -5):

$x = "Hello World!";

echo substr($x, -5, 3);

**Note** The last character has index -1.

## Negative Length

Use negative length to specify how many characters to omit, starting from the end of the string:

### Example

Get the characters starting from the "W" in "World" (index 5) and continue until 3 characters from the end.

Should end up with "Wor":

$x = "Hello World!";

echo substr($x, 5, -3);

# PHP - Escape Character

## Escape Character

To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes:

### Example

$x = "We are the so-called "Vikings" from the north.";

To fix this problem, use the escape character \":

### Example

$x = "We are the so-called \"Vikings\" from the north.";

## Escape Characters

Other escape characters used in PHP:

|  |  |  |
| --- | --- | --- |
| **Code** | **Result** |  |
| \' | Single Quote |  |
| \" | Double Quote |  |
| \$ | PHP variables |  |
| \n | New Line |  |
| \t | Tab |  |
|  |  |  |

# PHP ConstantsConstants are like variables, except that once they are defined they cannot be changed or undefined.

**PHP Constants**

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

**Note:** Unlike variables, constants are automatically global across the entire script.

## Create a PHP Constant

To create a constant, use the define() function.

### Syntax

define(*name*, *value*, *case-insensitive*);

Parameters:

* *name*: Specifies the name of the constant
* *value*: Specifies the value of the constant
* *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false. **Note:** Defining case-insensitive constants was deprecated in PHP 7.3. PHP 8.0 accepts only false, the value true will produce a warning.

### Example

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

define("GREETING", "Welcome to W3Schools.com!");

echo GREETING;

### Example

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

define("GREETING", "Welcome to W3Schools.com!", true);

echo greeting;

## PHP const Keyword

You can also create a constant by using the const keyword.

### Example

Create a constant with the const keyword:

const MYCAR = "Volvo";

echo MYCAR;

**const vs. define()**

* const cannot be created inside another block scope, like inside a function or inside an if statement.
* define can be created inside another block scope.

**PHP Operators**

PHP has different types of operators for different operations. They are as follows:

**1. Arithmetic Operators**

Arithmetic operators are used to perform arithmetic operations.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| Addition | + | $x+$y |
| Subtraction | - | $x-$y |
| Multiplication | \* | $x\*$y |
| Division | / | $x/$y |
| Modulus | % | $x%$y |
| Exponentiation | \*\* | $x\*\*$y |

**2. Assignment Operators**

These operators are used to assign values to variables.

|  |  |
| --- | --- |
| **Name** | **Evaluated as** |
| = | a=b |
| += | a=a+b |
| -= | a=a-b |
| \*= | a=a\*b |
| /= | a=a/b |
| %= | a=a%b |

**3. Comparison Operators**

These operators are used to compare two values.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| Equal | == | $x==$y |
| Identical | === | $x===$y |
| Not equal | != | $x!=$y |
| Not equal | <> | $x<>$y |
| Not Identical | !=== | $x!===$y |
| Greater than | > | $x>$y |
| Less than | < | $x<$y |
| Greater than or equal to | >= | $x >= $y |
| Less than or equal to | <= | $x <= $y |
| Spaceship | <=> | $x <=> $y |

**4. PHP Increment/ Decrement Operators**

These operators are used to increment/ decrement variable’s value.

|  |  |
| --- | --- |
| **Name** | **Operator** |
| Pre-Increment | ++$x |
| Post-Increment | $x++ |
| Pre-decrement | - -$x |
| Post-decrement | $x- - |

**5. PHP Logical Operators**

These are the logical operators that combine conditional statements.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| And | and | $x and $y |
| Or | or | $x or $y |
| Xor | xor | $x xor $y |
| And | && | $x && $y |
| Or | || | $x || $y |
| Not | ! | !&x |

**6. PHP String Operators**

PHP has these two operators designed for strings.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| Concatenation | . | $text1 . $text2 |
| Concatenation Assignment | .= | $text1 .= $text2 |

**7. PHP Array Operators**

These Operators are used to compare arrays.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| Union | + | $x + $y |
| Equality | == | $x = $y |
| Identity | === | $x === $y |
| Inequality | != | $x != $y |
| Inequality | < > | $x <> $y |
| !== Non-Identity | !== | $x !== $y |

**8. PHP Conditional Operators**

These operators assign values to operands based on the outcome of a certain condition.

|  |  |  |
| --- | --- | --- |
| **Name** | **Operator** | **Example** |
| Ternary | ?: | $x = exp1 ? exp2 : exp3 |
| Null Coalescing | ?? | $x = exp1 ?? exp2 |

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

## PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

* if statement - executes some code if one condition is true
* if...else statement - executes some code if a condition is true and another code if that condition is false
* if...elseif...else statement - executes different codes for more than two conditions
* [switch](https://www.w3schools.com/php/php_switch.asp) statement - selects one of many blocks of code to be executed

## PHP - The if Statement

The if statement executes some code if one condition is true.

### Syntax

if (*condition*) { *code to be executed if condition is true*;  
}

### Example

Output "Have a good day!" if 5 is larger than 3:

if (5 > 3) {

echo "Have a good day!";

}

We can also use variables in the if statement:

### Example

Output "Have a good day!" if $t is less than 20:

$t = 14;

if ($t < 20) {

echo "Have a good day!";

}

## PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

### Syntax

if (*condition*) {  
  *code to be executed if condition is true;*  
} else {  
  *code to be executed if condition is false;*}

### Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

$t = date("H");

if ($t < "20") {

echo "Have a good day!";

} else {

echo "Have a good night!";

}

## PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

### Syntax

if (*condition*) {  
  *code to be executed if this condition is true;*} elseif (*condition*) {  
 *code to be executed if first condition is false and this condition is true;*} else {  
  *code to be executed if all conditions are false;*}

### Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

$t = date("H");

if ($t < "10") {

echo "Have a good morning!";

} elseif ($t < "20") {

echo "Have a good day!";

} else {

echo "Have a good night!";

}

## Short Hand If

To write shorter code, you can write if statements on one line.

### Example

One-line if statement:

$a = 5;

if ($a < 10) $b = "Hello";

echo $b

## Short Hand If...Else

if...else statements can also be written in one line, but the syntax is a bit different.

### Example

One-line if...else statement:

$a = 13;

$b = $a < 10 ? "Hello" : "Good Bye";

echo $b;

## Nested If

You can have if statements inside if statements, this is called nested if statements.

### Example

An if inside an if:

$a = 13;

if ($a > 10) {

echo "Above 10";

if ($a > 20) {

echo " and also above 20";

} else {

echo " but not above 20";

}

}

# PHP switch Statement

The switch statement is used to perform different actions based on different conditions.

## The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

### 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":

"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!";

}

## The break Keyword

When PHP reaches a break keyword, it breaks out of the switch block.

This will stop the execution of more code, and no more cases are tested.

The last block does not need a break, the block breaks (ends) there anyway.

**Warning:** If you omit the break statement in a case that is not the last, and that case gets a match, the next case will also be executed even if the evaluation does not match the case!

### Example

What happens if we remove the break statement from case "red"?

$favcolor is red, so the code block from case "red" is executed, but since it has no break statement, the code block from case "blue" will also be executed:

$favcolor = "red";

switch ($favcolor) {

case "red":

echo "Your favorite color is red!";

case "blue":

"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!";

}

## The default Keyword

The default keyword specifies the code to run if there is no case match:

### Example

If no cases get a match, the default block is executed:

$d = 4;

switch ($d) {

case 6:

echo "Today is Saturday";

break;

case 0:

echo "Today is Sunday";

break;

default:

echo "Looking forward to the Weekend";

}

The default case does not have to be the last case in a switch block:

### Example

Putting  the default block elsewhere than at the end of the switch block is allowed, but not recommended.

$d = 4;

switch ($d) {

default:

echo "Looking forward to the Weekend";

break;

case 6:

echo "Today is Saturday";

break;

case 0:

echo "Today is Sunday";

}

**Note:** If default is not the last block in the switch block, remember to end the default block with a break statement.

## Common Code Blocks

If you want multiple cases to use the same code block, you can specify the cases like this:

### Example

More than one case for each code block:

$d = 3;

switch ($d) {

case 1:

case 2:

case 3:

case 4:

case 5:

echo "The weeks feels so long!";

break;

case 6:

case 0:

echo "Weekends are the best!";

break;

default:

echo "Something went wrong";

}