PHP

PHP (Hypertext Preprocessor) is a server-side scripting language and an open-source language specifically designed for web development. PHP is used to create dynamic web pages, interact with databases, and develop web applications.

Features of PHP

- Easy to Learn
- Open Source
- Platform Independent - PHP can run on different operating systems

  like- Windows, Linux, and Mac.
- Database Support - PHP can connect and work with databases

  like - MySQL.
- Fast and Efficient - PHP executes very quickly on the server side.
- Frameworks and Libraries - PHP provides powerful tools

  like - Laravel, CodeIgniter, and Symfony to develop web applications easily.

Advantages of PHP

- Low Cost
- Flexibility - PHP can work easily with structured languages and database languages.
- Scalability - The ability to create and manage a large website

- Rich Documentation - study material is easily available
- Integration - The ability to connect PHP with other technologies like HTML, CSS, JavaScript, and databases.
- Real-Time Access - the ability to process and display data instantly.

A History of PHP

PHP (Hypertext Preprocessor) was created by **Rasmus Lerdorf** in **1994**. Initially, PHP was designed to manage Rasmus's personal homepage, which is why he named it **"Personal Home Page" (PHP)**. Over time, it evolved into a server-side scripting language and became a standard for web development.

Evolution of PHP

1994: Rasmus Lerdorf wrote PHP in the C programming language. Initially, it was a simple scripting tool used to make web pages dynamic.

1995: Rasmus made PHP open-source, which led to a large community of developers contributing to it. In the same year, the first public release of PHP, called **PHP/FI (Form Interpreter)**, was launched.

1997: PHP 3 was released, rewritten by **Zeev Suraski** and **Andi Gutmans**. In this version, the name was changed to **"PHP: Hypertext Preprocessor."**

2000: PHP 4 was launched, which integrated the **Zend Engine**. This was a major milestone for PHP as it significantly improved performance and scalability.

2004: PHP 5 was released, introducing better support for **Object-Oriented Programming (OOP)** and new features.

2015: PHP 7 was launched, bringing massive improvements in speed and performance. This was the biggest upgrade in PHP's history.

2020: PHP 8 was released, featuring the **JIT (Just-In-Time) Compiler**, new syntax features, and improved error handling.

PHP Syntax

Syntax refers to the rules and structure for writing PHP code. Syntax defines how PHP code is written and organized so that the PHP interpreter can understand and execute it correctly. Syntax includes rules such as how variables are declared, how functions are written, and how code is structured.

A PHP script starts with <?php Tag and ends with ?> Tag.

Example:-

```
<?php

echo "Hello, World!";

?>
```

PHP Tags

Tags are used to embed PHP code within an HTML file. PHP tags tell the PHP interpreter which part of the code is PHP and which part is HTML or other content.

Standard Tags

```
<?php ..... ?>
```

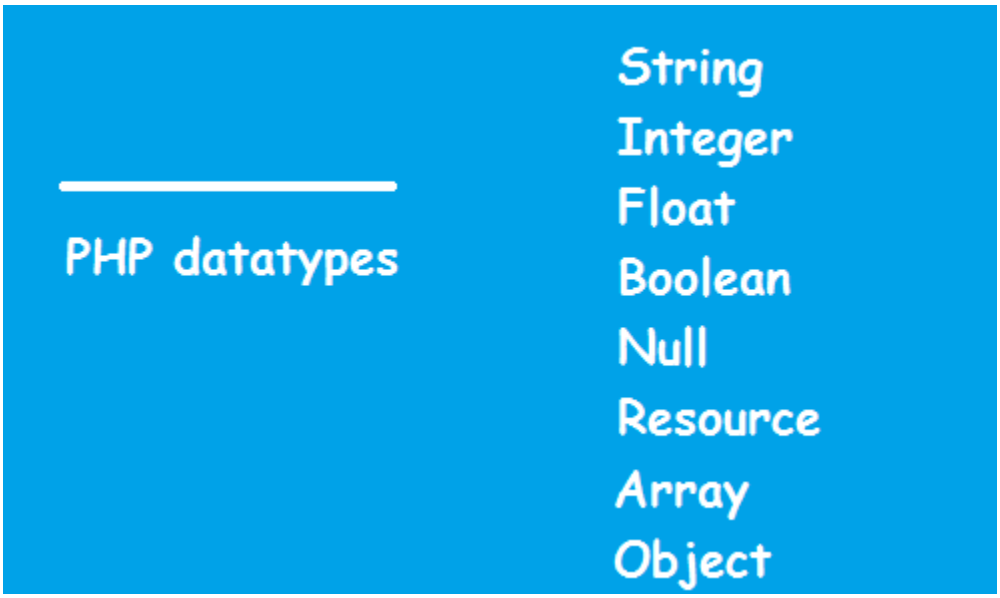
Short-Open Tags:

```
<? ..... ?>
```

PHP datatype

Data Types are used to store and manage different kinds of data. PHP is a **loosely typed language**, which means you don't need to explicitly declare the data type of a variable. PHP automatically determines the data type based on the value assigned.

PHP datatype



String

Definition: A string represents text data.

Example:-

```
$name = "Rahul";
$message = ‘welcome to howrah’;
```

Float

Definition: A float represents decimal or fractional numbers.

Example:-

```
$price = 99.99;
```

Boolean

Definition: A boolean represents two values: true or false.

Example:

```
$a = true;
```

$b = false;

ARRAY

An array is a data structure that allows you to store multiple values in a single variable. Arrays can hold elements of any data type, including integers, strings, objects, and even other arrays. PHP supports three types of arrays:

- **Indexed Arrays**
- **Associative Arrays**
- **Multidimensional Arrays**

Indexed Arrays

Indexed arrays use numeric indexes to access elements. The index starts at 0.

```
$fruits = array("Apple", "Banana", "Orange");

$fruits = ["Apple", "Banana", "Orange"];

echo $fruits[0]; // Outputs: Apple

echo $fruits[1]; // Outputs: Banana
```

Associative Arrays

Associative arrays use named keys that you assign to them.

```
$age = array(“name" => “karan”, “address" => “howrah”);

$age = [“name" => “pandey”, “mobile" => 7388847685];

echo $age[“name"]; // Outputs: karan

echo $age[“mobile"];   // Outputs: 7388847685
```

Multidimensional Arrays

Multidimensional arrays are arrays that contain one or more arrays.

```
$student=array(
        array("Apple", "Banana", "Orange"),
```

```
                array("Apple", "Banana", "Orange“));
echo $student[0][0];     //output   Apple
```

PHP Variables & Constants

Variables:-

A **variable** in programming is a container that

stores data or values.

Variables in PHP start with a $ symbol.

$A=“hello”

$B=123

PHP Variables & Constants

Constants:-

A **constant** in programming is a special type of variable

whose value cannot be changed once it is defined.

PHP, constants are defined using the define() function

```
define("A", "hello",);
```

OPERATOR

Operator is a symbol that performs operations on variables and values. Operators are used to manipulate data, perform calculations, compare values,

Types of OPERATOR

- Arithmetic operator
- Assignment operators
- Comparison operators
- Increment / Decrement operators
- Logical operators
- String operators

- Array operators
- Ternary Operator
- Bitwise operators

Arithmetic operator

Arithmetic operators are used to perform basic mathematical operations.

Operator	Name	Explanation	Example	Result
+	Addition	Adds two operands.	$a = 5 + 3;	8
-	Subtraction	Subtracts the second operand from the first.	$a = 5 - 3;	2
*	Multiplication	Multiplies two operands.	$a = 5 * 3;	15

Operator	Name	Explanation	Example	Result
/	Division	Divides the first operand by the second.	$a = 6 / 3;	2
%	Modulus	Returns the remainder of division.	$a = 5 % 3;	2
**	Exponentiation	Raises the first operand to the power of the second.	$a = 2 ** 3;	8

Assignment operators

Assignment operators are used to assign values to variables.

Assignment Operators Combined Table

Operator	Name	Explanation	Example	Result
=	Assignment	Assigns the value of the right operand to the left operand.	$a = 5;	5
+=	Add and Assign	Adds the right operand to the left operand and assigns	$a += 3;	8
-=	Subtract and Assign	Subtracts the right operand from the left operand.	$a = 3;	2
*=	Multiply and Assign	Fhg tee lhe tigH operand ieft_dthucesrge and assigns.	$a *= 3;	15
/=	Divide and Assign	Divides the left operand by the right operand and and,	$a /:= 3;	2
/=	Modulus and Assign	Takes the modulus of the left operand by the right operand.	$a *= 3;	2
**=	Exponentiation and Assign	Raises the left operand to the power of right operand and assigns.	$a *:= 3;	125

Comparison operators

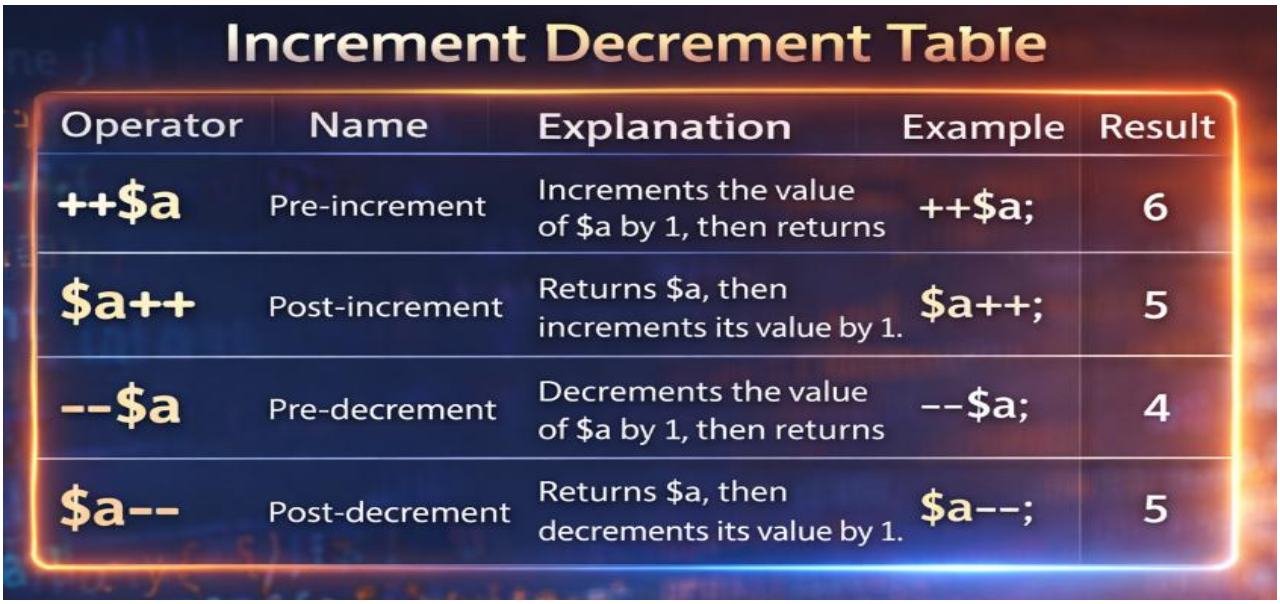
Comparison operators are used to compare two values.

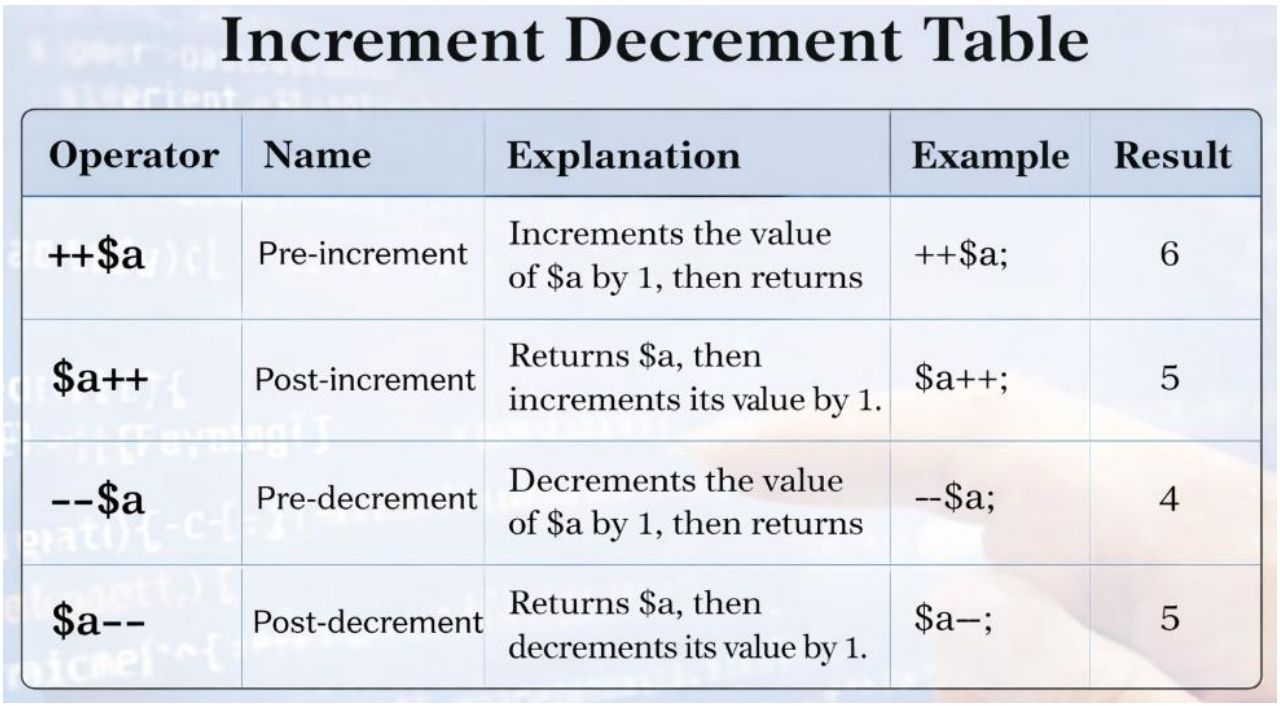
Comparison Operators Combined Table

Operator	Name	Explanation	Example	Result
==	Equal	Checks if two values are equal (type juggling allowed).	5 == "5";	true
===	Identical	Checks if two values are equal and of the same type.	5 === "5";	false
!=	Not Equal	Checks if two values are not equal then return true.	5 != 3;	true
<=	Not Equal	Same as !=.	5 <> 3;	true
!==	Not Identical	Checks if two values are not equal or not of the same type.	5 !== "5";	true
<	Less Than	Checks if the left operand is less than the right operand.	5 < 3;	false
>	Greater Than	Checks if the left operand is greater than the right operand.	5 > 3;	true
<=	Less Than or	Checks if the left operand is less than or equal to the right operanand.	5 <= 5;	True

Increment/ Decrement operators

These operators are used to increment or decrement a variable's value.





Logical operators

Logical operators are used to combine conditional statements.

Operator	Name	Explanation	Example	Result
and	And	True if both $x and $y are true	($x == 100 and $y == 50)	?
or	Or	True if either $x or $y is true	($x == 100 or $y == 80)	?
xor	Xor	True if either $x or $y is true, but not both	($x == 100 xor $y == 80)	?
&&	And	True if both $x and $y are true	($x == 100 && $y == 50)	?
||	Or	True if either $x or $y is true	($x == 100 || $y == 80)	?
!	Not	True if $x is not true	(!($x == 90))	?

String operators

PHP has two operators that are specially designed for strings.

String Operators Table

Operator	Name	Explanation	Example	Result
•	And	Concatenation of $txt1 and $txt2	$txt1 . . $txt2	42
.=	Or	True if either $x or $y is true	$txt1 . $txt2	True
=	Xor	Appends $txt2 to $txt1	$txt1 .=	True
&&	And	True if both $x and $y are true	$txt1 && $y = 50	True

Array operators

Array operators are used to compare arrays.

Array Operators Table

Operator	Name	Explanation	Example	Result
+	Union	Combines two arrays (duplicates are emoved).	[1, 2] + [2, 3];	[1, 2, 3]
==	Equality	Checks if two arrays have the same key/value pairs.	[1, 2] == [1, 2];	true
===	Identity	Checks if two arrays have the same key/value pairs in	[1, 2] == [1, 2];	true
!=	Inequality	Checks if two arrays are not equal	[1, 2] != [2, 3];	true
<>	Inequality	Same as !=.	[1, 2] <> [2, 3];	true
!==	Non-identity	Checks if two arrays are not identical.	[1, 2] !== [1, 2];	false

Ternary Operator

The ternary operator is a shorthand for if-else.

Ternary Operator Table

Operator	Name	Explanation	Example	Result
?:	Ternary	Returns expr1 if the condition is true, otherwise expr2.	$a = (5 > 3) ? "Yes" : "No";	
=	Equality	Checks if two arrays have the same key/value pairs in the same order and types.	[1, 2] = [1, 2]:	true

Bitwise operators

Used to perform bit-level operations.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1.
|	OR	Sets each bit to 1 if at least one of the bits is 1.
^	XOR	Sets each bit to 1 if only one of the bits is 1.
~	NOT	Inverts all the bits (changes 1 to 0 and 0 to 1).
<<	Left Shift	Shifts the bits of the left operand to the left by the number of positions specified by the right operand.
>>	Right Shift	Shifts the bits of the left operand to the right by the number of positions specified by the right operand.

Bitwise & AND operators

Sets a bit to 1 only if both corresponding bits are 1.

```
$a = 5;  // Binary: 0101
$b = 3;  // Binary: 0011
$result = $a & $b; // Binary: 0001
echo $result; // Output: 1
```

Bitwise | OR operators

Sets a bit to 1 if at least one of the corresponding bits is 1

```
$a = 5;  // Binary: 0101
$b = 3;  // Binary: 0011
$result = $a | $b;  // Binary: 0111 (Decimal: 7)
echo $result;  // Output: 7
```

Bitwise ^ XOR operators

Sets a bit to 1 if only one of the corresponding bits is 1.

```
$a = 5;  // Binary: 0101
$b = 3;  // Binary: 0011
$result = $a ^ $b;    // Binary: 0110 (Decimal: 6)
echo $result;  // Output: 6
```

Bitwise ~ NOT operators

Inverts all the bits (changes 1 to 0 and 0 to 1).

```
$a = 5;  // Binary: 0101
$b = 3;  // Binary: 0011
$result = $a ^ $b;  // 11111111111111111010
(On 32-bit systems)
echo $result;  // Output: 6
```

Bitwise << Left Shift operators

Shifts the bits of the left operand to the left by the number of positions specified by the right operand. Zeros are added to the right.

```
$a = 5;  // Binary: 0101
$b = 3;  // Binary: 0011
$result = $a ^ $b; // Binary: 1010 (Decimal: 10)
echo $result;  // Output: 10
```

Bitwise >> Right Shift operators

Shifts the bits of the left operand to the right by the number of positions specified by the right operand. Zeros are added to the left.

```
$a = 5;  // Binary: 0101
$result = $a >> 1; | Binary: 0010 (Decimal: 2)
echo $result; // Output: 2
```

Working of PHP