PHP Basic to Advance

VAISHNAVI JOSHI

Introduction to PHP

- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ”.php”

Introduction to PHP

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

Syntax

A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>

Example :-

```
<?php
        echo “Hello World” ;
 ?>
```

PHP statements end with a semicolon (;)
echo is used for Print data

PHP Variables

A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total_volume).

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)

PHP Variables Scope

In PHP, you can create (declare) variables anywhere in your code.
But where you create them affects where you can use them — this is called scope.

PHP has 3 types of scope:

Local – A variable made inside a function. You can only use it inside that function.

Global – A variable made outside functions. You can use it anywhere, but inside functions, you must use the global keyword.

Static – A special variable inside a function that keeps its value even after the function ends.

PHP echo and print Statements

echo and print are used to display output in PHP.

echo is faster and can output multiple values.

echo can use commas to separate values: echo "Hello", " World!";

print can only take one value: print "Hello World!";

Both are almost the same; echo is used more often.

Example:

```
echo "Hello World!";

print "Hello World!";
```

PHP Data Types

Variables hold different types of data. Each type is used for different kinds of values.

PHP supports these data types:

String – Text, like "Hello"

Integer – Whole numbers, like 100

Float (Double) – Decimal numbers, like 10.5

Boolean – True or false: true, false

Array – A list of values: [1, 2, 3]

Object – Data from a class (used in OOP)

NULL – No value at all

Resource – Special type for links to files or databases

PHP Constants

Constants are like variables, but their value can't change once set.

They are used to store fixed values (like settings or limits).

Constants do not use $ before their name.

Constant names must start with a letter or underscore.

Constants are automatically global, available anywhere in the script.

How to Create a Constant:

```
define("SITE_NAME", "MyWebsite");

echo SITE_NAME;   // Outputs: MyWebsite
```

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

PHP Operators

Operators are symbols used to perform operations on variables and values.

PHP divides operators into these groups:

Arithmetic Operators – Perform math: +, -, *, /, %

Assignment Operators – Assign values: =, +=, -=

Comparison Operators – Compare values: ==, !=, >, <, >=, <=

Increment/Decrement Operators – Increase or decrease by 1: ++, --

Logical Operators – Perform logical operations: &&, ||, !

String Operators – Concatenate strings: ., .=

Array Operators – Work with arrays: +, ==, ===

Conditional Assignment Operators – Assign if condition is true: ?: (ternary operator)

PHP Conditional Statements

Conditional statements allow you to run different actions based on different conditions.

PHP provides these conditional statements:

1. **if statement**
   Executes code if a condition is **true**.

2. **if...else statement**
   Runs one block of code if the condition is **true**, and another if the condition is **false**.

3. **if...elseif...else statement**
   Checks multiple conditions and executes the code for the **first true condition**.

4. **switch statement**
   Compares a variable to several possible values and runs the code for the matching value.

PHP switch Statement

The **switch** statement lets you choose from many options based on a specific condition.

How it works:

1. **Evaluate the expression** once.
2. **Compare the value** of the expression with each **case** value.
3. If a **match** is found, the corresponding block of code is executed.
4. The **break** keyword ends the switch statement (so it doesn't run other cases).
5. If no match is found, the **default** block is executed.

PHP Loops

Loops allow you to repeat the same block of code multiple times based on a condition.

PHP has the following types of loops:

1. **while**
   Runs the code as long as the condition is **true**.

2. **do...while**
   Runs the code **once**, then repeats as long as the condition is **true**.

3. **for**
   Runs the code a **specified number of times**.

4. **foreach**
   Loops through each **element** in an array and executes the code for each.

PHP Loops Example

1. while loop

```
$x = 1;
while ($x <= 3) {
  echo $x; // Outputs 123
  $x++;
}
```

2. do...while loop

```
$x = 1;
do {
  echo $x; // Outputs 123
  $x++;
} while ($x <= 3);
```

3. for loop

```
for ($x = 1; $x <= 3; $x++) {
  echo $x; // Outputs 123
}
```

4. foreach loop

```
$array = [1, 2, 3];
foreach ($array as $value) {
  echo $value; // Outputs 123
}
```

PHP Function

A **function** is a block of code that can be called to perform a task. You can use functions to avoid writing the same code multiple times.

How functions work:

1. **Define a function** with a name.
2. **Call the function** whenever you need it.

Exmaple :-

```
function sayHello($name) {
  echo "Hello, " . $name;
}
 sayHello("John"); // Outputs: Hello, John
```

Types of Functions in PHP

1. Predefined Functions (Built-in Functions)

These are functions that PHP provides by default. You don't need to define them, just call them.

Example: echo(), strlen(), array_merge()

2. User-Defined Functions

These are functions you define yourself to perform specific tasks in your code.

Example :- function greet($name) {

```
  echo "Hello, " . $name;
}
```

PHP Arrays and Types

Arrays in PHP are used to store multiple values in a single variable.

Types of Arrays:

1. Indexed Arrays

These arrays use numeric indexes (starting from 0).

Example:

```
$fruits = ["Apple", "Banana", "Cherry"];
```

2. Associative Arrays

These arrays use named keys to access values instead of numeric indexes.

Example:

```
$person = ["name" => "John", "age" => 30];
```

PHP Arrays and Types

3. Multidimensional Arrays

Arrays that contain other arrays, allowing for more complex data storage.

Example:

```
$students = [
  ["name" => "John", "age" => 20],
  ["name" => "Jane", "age" => 22]
];
```

PHP Global Variables

Superglobal variables in PHP are built-in variables that are always accessible, no matter where you are in your script—inside functions, classes, or files. You don't need to do anything special to access them.

They are **global** by nature and automatically available throughout your script.

PHP Form Handling

Create a Form: An HTML form is created with fields that ask for user input, such as name, email, etc.

Submit the Form: When the user submits the form, the form data is sent to a PHP script for processing (using either GET or POST).

PHP Processes the Data: PHP accesses the form data through $_GET (for GET method) or $_POST (for POST method), and then it can process or display it.

Return a Response: After processing, PHP can either show a confirmation message or store the data in a database.

GET vs POST in PHP Form Handling

When you submit a form in PHP, there are two main ways to send data to the server: **GET** and **POST**. These are called **HTTP methods**.

GET Method:

- The **GET** method sends form data **in the URL**.
- It's mainly used for **getting** or **retrieving** data.
- **Visible in the URL**: The data is shown in the browser's address bar.
- It’s **less secure** because the data is visible in the URL.
- Suitable for **non-sensitive data** or when you want the data to be shareable

GET vs POST in PHP Form Handling

POST Method:

- The **POST** method sends form data **in the background**, not in the URL.
- It's mainly used for **sending** data to be processed (like submitting a contact form).
- **Invisible in the URL**: Data is not visible to users.
- It’s **more secure** because sensitive information (like passwords) is not exposed in the URL.
- Ideal for **sensitive data** or when submitting large amounts of data.

Summary:

- **GET**: Data is visible in the URL, used for retrieving non-sensitive data.
- **POST**: Data is not visible in the URL, used for submitting sensitive or large data.

Thank You ☺