# ✅ Hello World in PHP

### **📌 Concept: Basic PHP Script**

PHP files use the .php extension and must have PHP tags:

<?php

// your PHP code here

?>

### **🔸 Example 1: Hello World**

#### **📄 hello.php**

<?php

echo "Hello, World!";

?>

### **🧠 Explanation:**

* <?php — starts the PHP code
* echo — used to output text or variables to the screen
* "Hello, World!" — the string we want to print
* ; — every PHP statement ends with a semicolon

### **▶️ How to Run:**

#### **Option 1: Using Built-in PHP Server**

If PHP is installed on your system

php -S localhost:8000

Place hello.php in the folder where you run the above command, and open:

http://localhost:8000/hello.php

#### **Option 2: Using CLI (Command Line)**

php hello.php

### **📝 Notes for You:**

📄 PHP Hello World Program

- PHP files end with `.php`

- Use `<?php ... ?>` to write PHP code

- Use `echo` to print text

- Every line must end with `;`

✅ Example:

<?php

echo "Hello, World!";

?>

# 

# 

# **✅ PHP Syntax: Variables, Data Types, Functions, Loops, Arrays, Conditionals**

## **1. 🔹 Variables**

### **📌 Syntax:**

$variableName = value;

* Variables start with $
* No need to declare data type
* Case-sensitive ($name ≠ $Name)

### **✅ Example:**

<?php

$name = "Aditya";

$age = 21;

echo "My name is $name and I am $age years old.";

?>

## **2. 🔹 Data Types in PHP**

PHP is dynamically typed but supports:

* String
* Integer
* Float
* Boolean
* Array
* Object
* NULL

### **✅ Example:**

<?php

$str = "Hello";

$int = 10;

$float = 10.5;

$bool = true;

$nullValue = null;

echo gettype($str); // Output: string

?>

## **3. 🔹 Conditionals (if, else, elseif)**

### **✅ Example:**

<?php

$marks = 85;

if ($marks >= 90) {

echo "Grade A";

} elseif ($marks >= 75) {

echo "Grade B";

} else {

echo "Grade C";

}

?>

## **4. 🔹 Loops (for, while, foreach)**

### **✅ for loop:**

<?php

for ($i = 1; $i <= 5; $i++) {

echo "Number: $i<br>";

}

?>

### **✅ while loop:**

<?php

$x = 1;

while ($x <= 3) {

echo "x = $x<br>";

$x++;

}

?>

### **✅ foreach loop (used for arrays):**

<?php

$colors = ["red", "green", "blue"];

foreach ($colors as $color) {

echo "Color: $color<br>";

}

?>

## **5. 🔹 Arrays (Indexed + Associative)**

### **✅ Indexed Array:**

<?php

$fruits = ["apple", "banana", "cherry"];

echo $fruits[1]; // Output: banana

?>

### **✅ Associative Array:**

<?php

$person = ["name" => "Aditya", "age" => 21];

echo $person["name"]; // Output: Aditya

?>

## **6. 🔹 Functions**

### **📌 Define and call a function:**

<?php

function greet($name) {

return "Hello, $name!";

}

echo greet("Aditya"); // Output: Hello, Aditya!

?>

## **📝 Notes Summary for You**

📘 PHP Syntax Essentials

✅ Variables

- Start with $

- Dynamic types

- $name = "Aditya";

✅ Data Types

- String, Int, Float, Bool, Array, NULL

- gettype($var)

✅ Conditionals

if (condition) { ... } elseif (...) { ... } else { ... }

✅ Loops

- for ($i = 0; $i < 10; $i++) { ... }

- while (cond) { ... }

- foreach ($arr as $item) { ... }

✅ Arrays

- Indexed: $arr[0]

- Associative: $arr["key"]

✅ Functions

function name($param) { return ...; }

# **✅ PHP Superglobals: $\_GET, $\_POST, $\_REQUEST, $\_FILES**

### **🧠 What are Superglobals?**

Superglobals are **built-in PHP arrays** that are always accessible, anywhere in the script — no need to declare them.

## **🔹 $\_GET: Retrieve data sent via URL (Query Parameters)**

### **🔸 Use Case: Search User by Name**

### **✅ URL:**

http://localhost/search.php?name=adi

### **✅ search.php:**

<?php

if (isset($\_GET['name'])) {

$name = $\_GET['name'];

echo "You searched for: $name";

} else {

echo "No search query.";

}

?>

## 

## 

## 

## **🔹 $\_POST: Retrieve form data sent with POST method**

### **🔸 Use Case: Login form submission**

### **✅ login.html**

<form action="login.php" method="POST">

Username: <input type="text" name="username" />

Password: <input type="password" name="password" />

<button type="submit">Login</button>

</form>

### **✅ login.php**

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$user = $\_POST['username'];

$pass = $\_POST['password'];

// Dummy check

if ($user === "admin" && $pass === "1234") {

echo "Login successful!";

} else {

echo "Invalid credentials.";

}

}

?>

## 

## 

## 

## 

## **🔹 $\_REQUEST: Accepts both $\_GET and $\_POST**

$name = $\_REQUEST['name']; // works for both ?name= or form data

# 

# **✅ Form Handling (Real-World Example)**

### **🔸 Use Case: Contact Us Form**

### **✅ contact.html**

<form action="contact.php" method="POST">

Name: <input type="text" name="name" /><br>

Email: <input type="email" name="email" /><br>

Message: <textarea name="message"></textarea><br>

<button type="submit">Send</button>

</form>

### **✅ contact.php**

<?php

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$name = htmlspecialchars($\_POST['name']);

$email = htmlspecialchars($\_POST['email']);

$message = htmlspecialchars($\_POST['message']);

// save to database or send email (not shown here)

echo "Thank you, $name. Your message was received.";

}

?>

**🔐 Good Practice:**

* Always sanitize input (htmlspecialchars, trim)
* Validate email format using filter\_var()

## **📝 Notes Summary:**

wasm

CopyEdit

📘 PHP Superglobals & Form Handling

✅ $\_GET

- Gets data via URL: http://site/page.php?key=value

✅ $\_POST

- Gets data via form submission (hidden from URL)

✅ $\_REQUEST

- Works for both GET and POST

✅ Form Handling:

- Always sanitize inputs using htmlspecialchars, trim

- Use method="POST" for secure input

# **✅ File Upload (Real-World Use Case: Upload Profile Picture)**

### **🔸 Use Case: Upload User Profile Picture**

### **✅ upload.html**

html

CopyEdit

<form action="upload.php" method="POST" enctype="multipart/form-data">

Select image: <input type="file" name="profilePic" />

<button type="submit">Upload</button>

</form>

enctype="multipart/form-data" is **mandatory** for file uploads.

### **✅ upload.php**

Sure brother! Here's your upload.php file with **line-by-line comments** explaining everything clearly 👇

php

CopyEdit

<?php

// Check if the form was submitted using POST method

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Access the uploaded file via $\_FILES superglobal

$file = $\_FILES['profilePic'];

// Check if there was no error during file upload

if ($file['error'] === 0) {

// Get the original name of the uploaded file

$filename = $file['name'];

// Get the temporary location of the uploaded file

$tempPath = $file['tmp\_name'];

// Set the destination path where the file should be moved

$targetPath = "uploads/" . basename($filename);

// Get the file extension in lowercase

$fileType = strtolower(pathinfo($targetPath, PATHINFO\_EXTENSION));

// Define allowed file types (for security)

$allowedTypes = ['jpg', 'png', 'jpeg'];

// Check if the uploaded file's type is allowed

if (in\_array($fileType, $allowedTypes)) {

// Move the file from temp location to final location

if (move\_uploaded\_file($tempPath, $targetPath)) {

// If move was successful, confirm success

echo "File uploaded successfully!";

} else {

// If moving the file failed, show error

echo "Failed to move uploaded file.";

}

} else {

// If file type is not allowed, show error

echo "Only JPG, PNG, and JPEG files are allowed.";

}

} else {

// If there was an error during upload, show message

echo "Error uploading file.";

}

}

?>

### **📁 Folder structure:**

Make sure you have a folder named uploads/ and it has write permission.

✅ File Upload:

- Use $\_FILES to handle file data

- Use move\_uploaded\_file() to store it

- Validate file type and size

📌 Important:

<form method="POST" enctype="multipart/form-data"> for file uploads

# ✅ include vs require

### **🔹 Purpose:**

Used to **import** or **reuse** code from other PHP files — commonly for headers, footers, configs, DB connections, etc.

### **✅ Example: Using include**

#### **📄 header.php**

<h1>Welcome to My Website</h1>

<hr>

#### **📄 home.php**

<?php

// Include header content from another file

include 'header.php';

// Rest of the page

echo "This is the home page.";

?>

### **✅ include vs require**

<?php

// include: load a file; if not found, script continues with a warning

include 'config.php'; // good for optional files

// require: load a file; if not found, script stops with a fatal error

require 'db\_connection.php'; // good for critical files

echo "Script continues if include fails, but not if require fails.";

?>

# ✅ Error Handling in PHP

PHP has **4 main methods** of error handling:

* error\_reporting() – define what errors to report
* try { } catch { } – Exception handling
* trigger\_error() – manually raise custom errors
* set\_error\_handler() – custom error function

### **✅ Example: Basic Exception Handling**

<?php

// Define a function that might raise an exception

function divide($a, $b) {

// Check if divisor is zero

if ($b == 0) {

// Throw an exception with a custom message

throw new Exception("Division by zero is not allowed.");

}

// If everything is fine, return the result

return $a / $b;

}

try {

// Attempt to call the divide function

echo divide(10, 0); // Will throw exception

} catch (Exception $e) {

// Catch block handles any exception thrown

// $e->getMessage() returns the error message

echo "Error: " . $e->getMessage();

}

?>

### **✅ Example: Manual Error Triggering**

<?php

$age = -5;

// Check for invalid age

if ($age < 0) {

// Manually trigger a warning (custom error message)

// E\_USER\_WARNING is the level of the error

trigger\_error("Age cannot be negative", E\_USER\_WARNING);

}

?>

## 

## 

## 

## 

## **📝 Notes Summary for You:**

📘 include vs require

- include('file.php') → warning on error, script continues

- require('file.php') → fatal error on missing file, script stops

📘 Error Handling

✅ try { } catch { }

- Use to handle runtime exceptions

- throw new Exception("msg")

✅ trigger\_error("msg", E\_USER\_WARNING)

- Manually raise an error

✅ error\_reporting(E\_ALL)

- Enable all error messages

✅ set\_error\_handler()

- Define custom error handler function

# ✅ Login with Sessions and Cookies

### **💡 What you’ll learn:**

* Store logged-in user using $\_SESSION
* Use setcookie() to implement "Remember Me"
* Protect dashboard with session check
* Destroy session on logout

### **📄 login.php**

<?php

// Start session to allow session variables

session\_start();

// Dummy credentials (you'd fetch these from a DB in real apps)

$validUsername = "admin";

$validPassword = "1234";

// Check if form is submitted via POST

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = $\_POST['username'];

$password = $\_POST['password'];

// Match user input with valid credentials

if ($username === $validUsername && $password === $validPassword) {

// Store user in session (session lives until browser closes or expires)

$\_SESSION['user'] = $username;

// Check if user selected "Remember Me"

if (isset($\_POST['remember'])) {

// Set a cookie that stores username for 7 days

setcookie("user", $username, time() + (7 \* 24 \* 60 \* 60), "/");

}

// Redirect to dashboard

header("Location: dashboard.php");

exit();

} else {

echo "Invalid credentials!";

}

}

?>

<!-- Simple login form -->

<form method="POST">

Username: <input type="text" name="username" /><br>

Password: <input type="password" name="password" /><br>

<label><input type="checkbox" name="remember" /> Remember Me</label><br>

<button type="submit">Login</button>

</form>

### **📄 dashboard.php**

<?php

session\_start();

// Check if user is already logged in via session

if (!isset($\_SESSION['user'])) {

// If not in session, check if "remember me" cookie exists

if (isset($\_COOKIE['user'])) {

// If cookie exists, restore session from cookie

$\_SESSION['user'] = $\_COOKIE['user'];

} else {

// No session or cookie, redirect to login

header("Location: login.php");

exit();

}

}

// Welcome message using session variable

echo "Welcome, " . $\_SESSION['user'] . "!<br>";

// Logout link

echo "<a href='logout.php'>Logout</a>";

?>

### **📄 logout.php**

<?php

session\_start();

// Clear all session variables

session\_unset();

// Destroy the session

session\_destroy();

// Remove the "remember me" cookie by setting its expiry in past

setcookie("user", "", time() - 3600, "/");

// Redirect to login

header("Location: login.php");

exit();

?>

## 

## 

## 

## **🧠 How This Works (All in Code)**

* On login:  
  + Valid credentials → session created → optional cookie set
* On dashboard:  
  + First checks session
  + If session not found, checks cookie and restores session
* On logout:  
  + Session is destroyed, cookie is removed
* "Remember Me" lets the user stay logged in for 7 days using cookie even after browser closes

## **📝 Notes Summary:**

📘 Sessions and Cookies in PHP

✅ session\_start()

- Must be called at top before any HTML

- Enables use of $\_SESSION

✅ $\_SESSION['user'] = $username;

- Stores data in memory across pages

✅ setcookie("user", $value, expiryTime, "/");

- Stores data on user's browser (like "remember me")

- Must be called before any output

✅ session\_unset(), session\_destroy()

- Used for logout

✅ Real-World Use:

- Sessions for secure login state

- Cookies for "remember me" login persistence