

All-PHP

PHP ALL

everything you need to know about PHP

1. echo

`echo` is used to output data to the screen.

```
<?php echo "Hello, World!";?>
```

2. print

`print` is similar to `echo`, but it always returns 1.

```
<?php print "Hello, World!";?>
```

3. PHP Data Types

PHP supports several data types:

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL

```
<?php $string = "Hello";$int = 5;$float = 5.7;$bool = true;$array = array  
("apple", "banana", "cherry");?>
```

4. Comments

Comments in PHP can be single-line or multi-line.

```
<?php// This is a single-line comment/* This is a multi-line comment*/?>
```

5. Variables Scope

PHP has three types of variable scope: local, global, and static.







```
<?php $x = 10; // Global scopefunction test() { $x = 20; // Local scope echo $x;} test();echo $GLOBALS['x']; // Accessing global scope?>
```

6. Arithmetic Operation

PHP supports standard arithmetic operations like addition, subtraction, etc.

```
<?php $x = 10;$y = 6;echo $x + $y; // Additionecho $x - $y; // Subtractionecho $x * $y; // Multiplicationecho $x / $y; // Division?>
```

Arithmetic Operations Summary

-  (Addition): Adds two values.
-  (Subtraction): Subtracts one value from another.
-  (Multiplication): Multiplies two values.
-  (Division): Divides one value by another.
-  (Modulus): Returns the remainder of a division.
-  (Exponentiation): Raises one value to the power of another.

7. Math Functions

7.1. min() and max() Function

Finds the minimum and maximum values.

```
<?php echo min(1, 2, 3); // 1echo max(1, 2, 3); // 3?>
```

7.2. abs() Function

Returns the absolute value.

```
<?php echo abs(-6.7); // 6.7?>
```

7.3. round() Function

Rounds a floating-point number.

```
<?php echo round(3.6); // 4?>
```

7.4. rand() Function

Generates a random number.

```
<?php echo rand(1, 10);?>
```

8. Logical Operations

logical operations are used to combine multiple conditions and return a boolean value (true or false). These operations are mainly used in conditional statements like `if`, `while`, `for`, and `switch` to control the flow of the program based on multiple conditions.

```
<?php $x = 10;$y = 20;if ($x == 10 && $y == 20) { echo "Both conditions are true.";} ?>
```

Logical Operators Summary

- `&&` (AND): Both conditions must be true.
- `||` (OR): At least one condition must be true.
- `!` (NOT): Inverts the condition's truth value.
- `xor`: Only one condition can be true (not both).

9. Arrays

```
<?php $fruits = array("apple", "banana", "cherry");echo $fruits[0]; // Output: apple?>
```

short hand syntax

```
<?php $fruits = ["apple", "banana", "cherry"];echo $fruits[0]; // Output: apple?>
```

10. Associative Arrays

```
<?php $ages = array("Peter" => 35, "John" => 40);echo $ages["Peter"]; // Output: 35?>
```

short hand syntax

```
<?php $ages = ["Peter" => 35, "John" => 40];echo $ages["Peter"]; // Output: 35?>
```

11. PHP **const** Keyword and **define** Function

const defines a constant, and **define()** defines a constant functionally.

```
<?php const PI = 3.14;define("GREETING", "Hello, World!");echo PI;echo GREETING;?>
```

12. if statements

```
<?php $x = 10;if ($x > 5) { echo "x is greater than 5";} ?>
```

13. Switch

```
<?php $color = "red";switch ($color) { case "red": echo "Your favorite color is red!"; break; case "blue": echo "Your favorite color is blue!"; break; default: echo "Your favorite color is neither red nor blue!";} ?>
```

14. For Loop

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

15. While Loop

```
<?php $x = 0;while ($x <= 10) { echo "The number is: $x <br>"; $x++;} ?>
```

16. For Each Loop

```
<?php $colors = array("red", "green", "blue", "yellow"); foreach($colors as $value) { echo "$value <br>";} ?>
```

17. isset() & empty() Function

```
<?php $var = "Hello"; if (isset($var)) { echo "Variable is set";} if (empty($var)) { echo "Variable is empty";} ?>
```

18. PHP Superglobals

18.1. \$GLOBALS

Used to access global variables.

```
<?php $x = 10; function test() { echo $GLOBALS['x'];} test();?>
```

18.2. \$_SERVER

Contains information about headers, paths, and script locations.

```
<?php echo $_SERVER['PHP_SELF'];?>
```

18.3. \$_REQUEST

Used to collect form data.

```
<?php $name = $_REQUEST['name'];?>
```

18.4. \$_POST

```
<?php $name = $_POST['name'];?>
```

18.5. \$_GET

```
<?php $name = $_GET['name'];?>
```

18.6. \$_FILES

```
<?php echo $_FILES['file']['name'];?>
```

18.7. \$_ENV

```
<?php echo $_ENV['HOME'];?>
```

18.8. \$_COOKIE

```
<?php echo $_COOKIE['user'];?>
```

18.9. \$_SESSION

```
<?php session_start(); $_SESSION['user'] = "John"; echo $_SESSION['user']; ?>
```

19. Making Functions in PHP

```
<?php function greet($name) { echo "Hello, $name!"; } greet("John");?>
```

20. Defining a Class

```
<?php class Car { public $color; public $model; public function __construct($color, $model) { $this->color = $color; $this->model = $model; } } ?>
```

21. PHP `__construct` Function

A constructor allows you to initialize an object's properties upon creation of the object.

If you create a `__construct()` function, PHP will automatically call this function when you create an object from a class.

Notice that the construct function starts with two underscores (`__`)!

```
<?php class Car { public $color; public $model; public function __construct($color, $model) { $this->color = $color; $this->model = $model; } } $car1 = new Car("red", "BMW"); echo $car1->color; ?>
```

22. PHP `__destruct` Function

A destructor is called when the object is destructed or the script is stopped or exited.

If you create a `__destruct()` function, PHP will automatically call this function at the end of the script.

Notice that the destruct function starts with two underscores (`__`)!

The example below has a `__construct()` function that is automatically called when you create an object from a class, and a `__destruct()` function that is automatically called at the end of the script:

```
<?php class Car { public $color; public $model; public function __construct($color, $model) { $this->color = $color; $this->model = $model; } public function __destruct() { echo "The car is being destroyed."; } } ?>
```

23. PHP Access Modifiers

PHP provides three access modifiers:

- `public`
- `protected`
- `private`

23.1. Public

```
<?php class Car { public $model;} ?>
```

23.2. Protected

```
<?php class Car { protected $model;} ?>
```

23.3. Private

```
<?php class Car { private $model;} ?>
```

24. PHP Static Methods

Static methods can be called directly - without creating an instance of the class first.

```
<?php class Car { public static function hello() { echo "Hello!"; } } Car::hello();?>
```

25. PHP Static Properties

Static properties can be called directly - without creating an instance of a class.

```
<?php class Car { public static $model = "BMW";} echo Car::$model; ?>
```

26. include files

include files allow you to insert the contents of one PHP file into another. This is especially useful for reusing code such as headers, footers, or navigation menus across multiple web pages, improving code modularity and maintainability.

in short

- `include` : Includes a file, script continues even if the file is not found.
- `require` : Includes a file, but stops execution if the file is missing.
- `include_once` and `require_once` : Ensures a file is included only once.

1_header.php_ (to be reused across different pages):


```
<!-- This is header.php --> <h1>Welcome to My Website</h1>
```

2. `index.php` (main file where `header.php` is included):

```
<?php include 'header.php'; // Include the header file ?> <p>This is the homepage content.</p>
```

Output

```
Welcome to My Website This is the homepage content.
```

Advanced filters

`filter_var()` is a powerful function used to **validate** and **sanitize** data. It takes an input variable and applies a specific filter to it. This is useful for ensuring that user input is clean and secure, especially when working with forms, URLs, emails, or other external data.

Syntax

```
filter_var(variable, filter, options);
```

- **variable**: The input variable to be filtered.
- **filter**: The filter to apply (optional, defaults to `FILTER_DEFAULT`).
- **options**: Additional options or flags (optional).

Common Filters

1. **Sanitization Filters**: Modify the input to remove unwanted characters (e.g., stripping HTML tags).
2. **Validation Filters**: Check whether the input is valid (e.g., checking if an email is correctly formatted).

Examples

1. Sanitizing a String

Removes all HTML tags from the string.

```
<?php $str = "<h1>Hello, World!</h1>"; $sanitizedStr = filter_var($str, FILTER_SANITIZE_STRING); echo $sanitizedStr; // Output: Hello, World!>
```

2. Validating an Email Address

Checks if the given string is a valid email address.

```
<?php $email = "halo.info@example.com"; if (filter_var($email, FILTER_VALIDATE_EMAIL)) { echo "Valid email address.";} else { echo "Invalid email address.";} ?>
```

3. Validating an Integer

Checks if the input is a valid integer.

```
<?php $int = "123"; if (filter_var($int, FILTER_VALIDATE_INT)) { echo "Valid integer.";} else { echo "Invalid integer.";} ?>
```

4. Sanitizing a URL

Removes illegal characters from a URL.

```
<?php $url = "https://www.example.com/somepage"; $sanitizedUrl = filter_var($url, FILTER_SANITIZE_URL); echo $sanitizedUrl; // Output: https://www.example.com/somepage ?>
```

5. Validating a URL

Checks if the given string is a valid URL.

```
<?php $url = "https://www.example.com"; if (filter_var($url, FILTER_VALIDATE_URL)) { echo "Valid URL.";} else { echo "Invalid URL.";} ?>
```

Using Options with `filter_var()`

You can pass additional options with certain filters to customize their behavior.

Example: Validating an Integer with a Range:

```
<?php $int = 100; $options = array( "options" => array("min_range" => 1,"max_range" => 200) ); if (filter_var($int, FILTER_VALIDATE_INT, $options)) { echo "Valid integer within range.";} else { echo "Invalid integer or out of range.";} ?>
```

Common Filters

1. **FILTER_SANITIZE_STRING**: Removes tags and unwanted characters from a string.
2. **FILTER_SANITIZE_EMAIL**: Removes illegal characters from an email.
3. **FILTER_SANITIZE_URL**: Removes illegal characters from a URL.
4. **FILTER_VALIDATE_INT**: Validates if the input is an integer.
5. **FILTER_VALIDATE_BOOLEAN**: Validates if the input is a boolean.
6. **FILTER_VALIDATE_FLOAT**: Validates if the input is a float.
7. **FILTER_VALIDATE_URL**: Validates if the input is a URL.
8. **FILTER_VALIDATE_EMAIL**: Validates if the input is an email.

Summary