PHP Manual

Purpose: This manual explains core PHP concepts for beginners. Each chapter contains clear explanations and short, practical examples you can copy and run.

Table of Contents

15. Array Functions

16. Debugging PHP

17. Obtaining PHP

18. Types

19. Creating Extensions

20. Using Extensions

---

1. Language Reference

What it is: The language reference documents the syntax and constructs of PHP: how to write statements, define variables, loops, conditionals, functions, classes, and more.

Key points:

* PHP is a server-side scripting language primarily for web development.

* PHP scripts are executed on the server and typically return HTML to the client.

Example — basic script:

```
<?php
// This is a single-line comment
echo "Hello, world!"; // prints text to the output
?>
```

Run this by placing it in a `.php` file on a server with PHP installed.

2. Features

Highlights:

* Easy to learn syntax similar to C, Java, and Perl.
* Wide standard library for strings, arrays, files, sessions, and more.
* Built-in web features: form handling, cookies, sessions.
* Support for many databases (MySQL, PostgreSQL, SQLite).
* Extensible via extensions written in C.

Example — connecting to MySQLi:

```
<?php
$mysqli = new mysqli("localhost", "user", "pass", "dbname");
if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
}
echo "Connected successfully";
?>
```

3. Function Reference

What it is: A catalog of built-in functions (`strlen`, `array_push`, `json_encode`, etc.). Learn parameter lists, return values, and examples.

Tip: Use `function_exists('name')` to check if a function is available (useful when extensions may vary).

Example — `json_encode` and `json_decode`:

```php
<?php
$data = ['name' => 'Ana', 'age' => 30];
$json = json_encode($data);
echo $json; // {"name":"Ana","age":30}

$decoded = json_decode($json, true);
var_dump($decoded);
?>
```

4. Appendixes

What they contain: Extras such as configuration (php.ini), migration notes, encoding, supported platforms, and quick reference tables.

Example appendix topic — `php.ini` common settings:

* `display_errors = Off` (in production)
* `memory_limit = 128M`

* `upload_max_filesize = 8M`

5. Basic Syntax

PHP tags:

* Standard: `<?php ... ?>` (recommended)
* Short: `<? ... ?>` (disabled on some setups)
* Echo short: `<?= $var ?>` (prints value)

Statements: end with semicolon `;`.

Example — control flow:

```
<?php
$hour = date('H');
if ($hour < 12) {
    echo "Good morning";
} elseif ($hour < 18) {
    echo "Good afternoon";
} else {
    echo "Good evening";
```

```
}
?>
```

6. Variables

Basics:

* Variables start with `$` — e.g., `$name`.
* PHP is dynamically typed — a variable can hold any type.
* Variable names are case-sensitive.

Scope: local (inside function), global (outside), superglobals (`$_POST`, `$_GET`, `$_SESSION`, etc.).

Example — variables and superglobals:

```
<?php
$name = "Marko";
$age = 25;

// Using GET parameter ?page=home
$page = $_GET['page'] ?? 'home';
```

```
echo "Name: $name, Age: $age, Page: $page";
?>
```

7. Functions

Defining functions:

* Use `function name($arg) { ... }`.
* Can return values with `return`.

Example — simple function:

```
<?php
function greet($name) {
    return "Hello, " . htmlspecialchars($name);
}

echo greet("Ana <script>alert('x')</script>");
?>
```

This example uses `htmlspecialchars` to avoid XSS when printing user input.

8. Operators

Common operators:

* Arithmetic: `+`, `-`, `*`, `/`, `%`
* Assignment: `=`, `+=`, `-=`
* Comparison: `==`, `===` (strict), `!=`, `<`, `>`
* Logical: `&&`, `||`, `!`
* String concatenation: `.`

Example — strict vs. loose comparison:

```php
<?php
var_dump(0 == '0');   // true
var_dump(0 === '0');  // false (different types)
?>
```

9. Cookies

What they are: Small pieces of data stored on the user's browser. Set via `setcookie()` before any output.

Example — set and read a cookie:

```
<?php
// set a cookie for 7 days
setcookie('site_lang', 'en', time() + 7*24*60*60, '/');

// later, read it
$lang = $_COOKIE['site_lang'] ?? 'en';
echo "Language: $lang";
?>
```

Security tip: Do not put sensitive data in cookies. Use sessions on server-side for secure info.

10. PHP and HTML

Embedding PHP in HTML: PHP often generates HTML dynamically.

Example — simple template:

```
<!doctype html>
<html>
<head><meta charset="utf-8"><title>Welcome</title></head>
<body>
<?php $user = 'Jelena'; ?>
<h1>Welcome, <?= htmlspecialchars($user) ?>!</h1>
</body>
</html>
```

Note: Use `htmlspecialchars()` when inserting user data into HTML to prevent XSS.

11. Strings

Operations: concatenation with `.`, functions like `strlen`, `strpos`, `substr`, `str_replace`.

Example — manipulating strings:

```
<?php
$s = "Hello, World!";
echo strlen($s); // length
echo substr($s, 7, 5); // "World"
echo str_replace('World', 'PHP', $s); // "Hello, PHP!"
?>
```

Interpolation: double-quoted strings allow variable interpolation: `"Hello $name"`.

12. Arrays

Types: indexed arrays, associative arrays, multidimensional arrays.

Example — arrays:

```
<?php
$fruits = ['apple', 'banana', 'orange'];
```

```
$person = ['name' => 'Milan', 'age' => 40];


echo $fruits[1]; // banana
echo $person['name']; // Milan


// multidimensional
$matrix = [[1,2],[3,4]];
echo $matrix[1][0]; // 3
?>
```

13. Functions (advanced)

Variable functions: call function by name in a variable.

```
<?php
function sayHi() { echo "Hi"; }
$fn = 'sayHi';
$fn(); // calls sayHi()
?>
```

Anonymous functions / Closures:

```php
<?php
$greet = function($name) { return "Hi, $name"; };
echo $greet('Sara');

// closure capturing variable by reference
$count = 0;
$inc = function() use (&$count) { $count++; };
$inc();
echo $count; // 1
?>
```

14. Classes and Objects

OOP basics: class, properties, methods, constructor, visibility (`public`, `protected`, `private`).

Example — simple class:

```php
<?php
class Car {
    public $make;
```

```
    private $speed = 0;

    public function __construct($make) {
        $this->make = $make;
    }

    public function accelerate($amount) {
        $this->speed += $amount;
    }

    public function getSpeed() {
        return $this->speed;
    }
}

$car = new Car('Toyota');
$car->accelerate(50);
echo $car->getSpeed(); // 50
?>
```

Inheritance and interfaces are next steps: `extends` and `implements`.

15. Array Functions

Useful functions: `array_map`, `array_filter`, `array_reduce`, `array_merge`, `in_array`, `array_key_exists`.

Example — `array_map` and `array_filter`:

```
<?php
$nums = [1,2,3,4,5];
$squared = array_map(fn($n) => $n*$n, $nums);
$even = array_filter($nums, fn($n) => $n % 2 === 0);
print_r($squared);
print_r($even);
?>
```

16. Debugging PHP

Techniques:

* `var_dump()` and `print_r()` for inspecting values.

* `error_reporting(E_ALL)` and `ini_set('display_errors', 1)` for development.

* Use Xdebug for step-debugging.

Example — enable errors (development only):

```
<?php
ini_set('display_errors', 1);
error_reporting(E_ALL);

// cause a warning
echo $undefinedVar;
?>
```

Tip: Never display errors in production — log them instead using `error_log()` or a logging library.

17. Obtaining PHP

Ways to get PHP:

* Use OS packages (APT/YUM) e.g., `apt install php`.

* Download from php.net and compile.

* Use pre-built stacks (XAMPP, MAMP, Laragon) for local development.

Quick local setup (Linux):

sudo apt update

sudo apt install php php-cli php-mbstring php-xml php-mysql

18. Types

Scalar types: `int`, `float`, `string`, `bool`.

Compound types: `array`, `object`, `callable`, `iterable`.

Special types: `resource`, `null`.

Type declarations: PHP supports type hints and return types since 7.x.

```php
<?php
function sum(int $a, int $b): int {
    return $a + $b;
}

echo sum(2,3); // 5
?>
```

Nullable types: `?Type` allows `null`.

19. Creating Extensions

Overview: Extensions are native modules written in C that add functions, classes, or performance improvements.

When to create:

* Need very high performance.
* Wrap existing C libraries.

Short outline:

1. Install PHP source and development headers.
2. Use `ext_skel` to generate skeleton.
3. Implement functions in C, register them.
4. Compile and enable in `php.ini`.

Note: Creating extensions requires C knowledge — start with userland PHP first.

---

20. Using Extensions

Common extensions: `pdo_mysql`, `mysqli`, `gd` (images), `mbstring`, `curl`, `json`, `openssl`.

Enable/disable: via `php.ini` or package manager. Use `php -m` to list loaded modules.

Example — using PDO:

```
<?php
try {
```

```
    $pdo = new PDO('mysql:host=127.0.0.1;dbname=test',
'user', 'pass');
    $stmt = $pdo->query('SELECT name FROM users');
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
        echo $row['name'] . "\n";
    }
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
?>
```

Closing notes

This manual is a starting point. For complete reference and the latest changes, consult the official PHP documentation at [php.net]. Practice by building small projects (guestbook, todo app, blog) and read the manual entries for each feature you use.

Good luck — keep experimenting!