PHP Quick Reference

Syntax, arrays, OOP, database, file I/O essentials

Basics

Hello World

```
<?php
echo "Hello, World!\n";
// PHP code must be inside <?php ... ?> tags
```

Run PHP

```
php script.php          # run a file
php -r 'echo "hi\n";'  # run inline code
php -S localhost:8000   # built-in dev server
```

Comments

```
// single-line comment
# also single-line
/* multi-line
   comment */
```

Variables & Types

Variables

```
$name = "PHP";          // string
$version = 8.3;         // float
$count = 42;            // int
$active = true;         // bool
$items = null;          // null
```

Type Checking

`gettype($x)`	Returns type as string
`is_string($x)`	Check if string
`is_int($x)`	Check if integer
`is_array($x)`	Check if array
`is_null($x)`	Check if null
`isset($x)`	Check if set and not null
`empty($x)`	Check if empty (falsy)

Type Casting

```
$n = (int) "42";       // 42
$s = (string) 3.14;    // "3.14"
$b = (bool) "";        // false
$a = (array) $obj;     // object to array
```

Constants

```
define("MAX_SIZE", 100);
const API_VERSION = "v2";
echo MAX_SIZE;   // 100
```

Strings

String Basics

```
$name = "World";
echo "Hello, $name!";         // variable interpolation
echo 'Hello, $name!';         // literal (no interpolation)
echo "Value: {$arr['key']}";  // complex expression
```

String Functions

`strlen($s)`	String length in bytes
`mb_strlen($s)`	String length in characters (multibyte-safe)
`strtolower($s)`	Convert to lowercase
`strtoupper($s)`	Convert to uppercase
`trim($s)`	Strip whitespace from both ends
`str_replace(a, b, $s)`	Replace **a** with **b** in **$s**
`substr($s, 0, 5)`	Substring from position 0, length 5
`strpos($s, 'find')`	Find position of substring (false if not found)
`explode(',', $s)`	Split string into array
`implode(',', $a)`	Join array into string

Heredoc & Nowdoc

```
$html = <<<HTML
<p>Hello, $name</p>
HTML;
$raw = <<<'TEXT'
No $interpolation here
TEXT;
```

Arrays

Indexed & Associative

```
$nums = [1, 2, 3];                  // indexed
$user = ["name" => "Alice", "age" => 30];  // associative
$nums[] = 4;                         // append
echo $user["name"];                  // access
```

Array Functions

`count($a)`	Number of elements
`array_push($a, $v)`	Append to end
`array_pop($a)`	Remove and return last element
`array_merge($a, $b)`	Merge two arrays
`in_array($v, $a)`	Check if value exists
`array_key_exists($k, $a)`	Check if key exists
`array_map($fn, $a)`	Apply function to each element
`array_filter($a, $fn)`	Filter elements by callback
`sort($a)`	Sort in place (reindexes)
`array_keys($a)`	Return all keys

Iteration

```
foreach ($users as $user) { echo $user; }
foreach ($map as $key => $value) {
    echo "$key: $value\n";
}
```

Functions

Basic Function

```
function add(int $a, int $b): int {
    return $a + $b;
}
echo add(3, 5);
```

Default & Named Arguments

```
function greet(string $name, string $greeting = "Hello"): string {
    return "$greeting, $name!";
}
greet("Alice");
greet(greeting: "Hi", name: "Bob");  // named args (PHP 8+)
```

Arrow Functions

```
$double = fn(int $x): int => $x * 2;
$nums = array_map(fn($n) => $n * 10, [1, 2, 3]);
```

Closures

```
$factor = 3;
$multiply = function(int $x) use ($factor): int {
    return $x * $factor;
};
echo $multiply(5);   // 15
```

Classes & Objects

Class Definition

```
class User {
    public function __construct(
        private string $name,
        private int $age = 0,
    ) {}
    public function greet(): string { return "Hi, {$this->name}"; }
}
```

Inheritance & Interfaces

```
interface Printable {
    public function toString(): string;
}
class Admin extends User implements Printable {
    public function toString(): string { return "Admin"; }
}
```

Visibility

`public`	Accessible from anywhere
`protected`	Accessible from class and subclasses
`private`	Accessible only within the class
`readonly`	Can only be assigned once (PHP 8.1+)
`static`	Belongs to the class, not instances
`abstract`	Must be implemented by subclass

Traits

```
trait Timestamped {
    public function createdAt(): string {
        return date('Y-m-d H:i:s');
    }
}
class Post { use Timestamped; }
```

Error Handling

Try / Catch / Finally

```
try {
    $result = riskyOperation();
} catch (InvalidArgumentException $e) {
    echo "Bad input: " . $e->getMessage();
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
} finally { cleanup(); }
```

Custom Exceptions

```
class ApiException extends RuntimeException {
    public function __construct(string $message, private int
$statusCode = 500) {
        parent::__construct($message, $statusCode);
    }
}
```

Null Safety (PHP 8+)

```
$len = $user?->address?->zip;  // nullsafe operator
$name = $input ?? "default";   // null coalescing
$data ??= [];                  // null coalescing assignment
```

PHP Quick Reference

File I/O

Read & Write Files

```
$content = file_get_contents("data.txt");
file_put_contents("out.txt", $content);
$lines = file("data.txt", FILE_IGNORE_NEW_LINES);
```

File Handle

```
$f = fopen("log.txt", "a");
fwrite($f, "entry\n");
fclose($f);
```

File Functions

`file_exists($path)`	Check if file exists
`is_dir($path)`	Check if path is a directory
`mkdir($path, 0755, true)`	Create directory recursively
`unlink($path)`	Delete a file
`glob('*.txt')`	Find files matching pattern
`realpath($path)`	Resolve full absolute path

Database

PDO Connection

```
$pdo = new PDO(
    "mysql:host=localhost;dbname=app",
    "user", "password",
    [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
);
```

Prepared Statements

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");
$stmt->execute([":id" => 42]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);
```

Insert & Update

```
$stmt = $pdo->prepare("INSERT INTO users (name, email) VALUES
(?, ?)");
$stmt->execute(["Alice", "alice@example.com"]);
$id = $pdo->lastInsertId();
```

PDO Fetch Modes

`fetch()`	Fetch single row
`fetchAll()`	Fetch all rows
`FETCH_ASSOC`	Return as associative array
`FETCH_OBJ`	Return as anonymous object
`FETCH_CLASS`	Return as instance of specified class

Common Functions

JSON

```
$json = json_encode(["name" => "Alice", "age" => 30]);
$data = json_decode($json, true);  // true = assoc array
$data = json_decode($json);        // object
```

Date & Time

```
echo date("Y-m-d H:i:s");           // 2026-03-26 12:00:00
$ts = strtotime("+1 week");
$dt = new DateTime("2026-01-01");
echo $dt->format("D, M j");          // Thu, Jan 1
```

Math & Random

`abs($n)`	Absolute value
`round($n, 2)`	Round to 2 decimal places
`ceil($n) / floor($n)`	Round up / down
`min($a, $b) / max($a, $b)`	Minimum / maximum
`random_int(1, 100)`	Cryptographically secure random int
`number_format($n, 2)`	Format with thousands separator

Regular Expressions

```
preg_match('/^[a-z]+$/i', $str, $matches);
preg_match_all('/\d+/', $str, $all);
$result = preg_replace('/\s+/', ' ', $str);
```