PHP クイックリファレンス

構文、配列、OOP、データベース、ファイル I/O の基本

基本

Hello World

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

PHP の実行

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

コメント

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

変数と型

変数

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

型チェック

`gettype($x)`	型を文字列で返す
`is_string($x)`	文字列か確認
`is_int($x)`	整数か確認
`is_array($x)`	配列か確認
`is_null($x)`	null か確認
`isset($x)`	セットされていて null でないか確認
`empty($x)`	空（falsy）か確認

型キャスト

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

定数

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

文字列

文字列の基本

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

文字列関数

`strlen($s)`	バイト単位の文字列長
`mb_strlen($s)`	文字単位の文字列長（マルチバイト対応）
`strtolower($s)`	小文字に変換
`strtoupper($s)`	大文字に変換
`trim($s)`	両端の空白を削除
`str_replace(a, b, $s)`	`$s` 内の `a` を `b` に置換
`substr($s, 0, 5)`	位置 0 から長さ 5 の部分文字列
`strpos($s, 'find')`	部分文字列の位置を検索（見つからない場合は false）
`explode(',', $s)`	文字列を配列に分割
`implode(',', $a)`	配列を文字列に結合

ヒアドキュメントと Nowdoc

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

配列

インデックス配列と連想配列

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

配列関数

`count($a)`	要素数
`array_push($a, $v)`	末尾に追加
`array_pop($a)`	末尾の要素を削除して返す
`array_merge($a, $b)`	2 つの配列をマージ
`in_array($v, $a)`	値が存在するか確認
`array_key_exists($k, $a)`	キーが存在するか確認
`array_map($fn, $a)`	各要素に関数を適用
`array_filter($a, $fn)`	コールバックで要素をフィルタリング
`sort($a)`	インプレースでソート（再インデックス）
`array_keys($a)`	全キーを返す

反復処理

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

関数

基本的な関数

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

デフォルト値と名前付き引数

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

アロー関数

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

クロージャ

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

クラスとオブジェクト

クラスの定義

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

継承とインターフェース

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

可視性

`public`	どこからでもアクセス可能
`protected`	クラスとサブクラスからアクセス可能
`private`	クラス内からのみアクセス可能
`readonly`	一度だけ代入可能（PHP 8.1+）
`static`	インスタンスではなくクラスに属する
`abstract`	サブクラスで実装必須

トレイト

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

エラー処理

Try / Catch / Finally

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

カスタム例外

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

Null 安全（PHP 8+）

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

ファイル I/O

ファイルの読み書き

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

ファイルハンドル

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

ファイル関数

`file_exists($path)`	ファイルが存在するか確認
`is_dir($path)`	パスがディレクトリか確認
`mkdir($path, 0755, true)`	ディレクトリを再帰的に作成
`unlink($path)`	ファイルを削除
`glob('*.txt')`	パターンに一致するファイルを検索
`realpath($path)`	完全な絶対パスを解決

データベース

PDO 接続

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

プリペアドステートメント

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

挿入と更新

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

PDO のフェッチモード

`fetch()`	1 行取得
`fetchAll()`	全行取得
`FETCH_ASSOC`	連想配列として返す
`FETCH_OBJ`	匿名オブジェクトとして返す
`FETCH_CLASS`	指定クラスのインスタンスとして返す

よく使う関数

JSON

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

日付と時刻

```
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
```

数学とランダム

`abs($n)`	絶対値
`round($n, 2)`	小数点以下 2 桁に丸め
`ceil($n) / floor($n)`	切り上げ / 切り捨て
`min($a, $b) / max($a, $b)`	最小値 / 最大値
`random_int(1, 100)`	暗号学的に安全なランダム整数
`number_format($n, 2)`	千の区切り付きでフォーマット

正規表現

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