SkillNewton

PHP 8.3 Cheat Sheet

PHP 8.3 syntax on one page: types, strings, arrays, match, functions, OOP, forms, sessions, PDO, password hashing and errors.

> This PHP cheat sheet summarises the syntax and built-in functions you need to write modern PHP 8.3, with a working snippet on every row. It suits students learning PHP for web development and developers coming back to the language after PHP 5 or 7. It covers types, arrays, `match`, OOP features such as enums and readonly properties, and the secure basics: PDO prepared statements, password hashing and sessions. Run any example with `php file.php`.

How to use it

1. **Check your version.** Run `php -v`. Everything here works on PHP 8.3; a few rows note the version that introduced a feature.
2. **Turn on strict types.** Add `declare(strict_types=1);` at the top of each file so type mistakes fail loudly.
3. **Use the secure patterns.** Copy the PDO, password and session sections as they are; they are the parts beginners most often get wrong.
4. **Look up details.** Search a function name on php.net to see every parameter and its return type.

Syntax basics

`<?php`	Opens PHP code. Omit the closing `?>` in files that contain only PHP.
`$name = 'Asha';`	Variables start with `$` and are case-sensitive.
`echo "Hello, $name";`	Outputs text. Double quotes interpolate variables; single quotes do not.
`const MAX_SEATS = 30;`	A constant.
`// comment # comment /* block */`	Comments.
`<?= htmlspecialchars($title) ?>`	Short echo tag for templates, with output escaping.
`require __DIR__ . '/config.php';`	Includes a file, stopping with an error if missing. `require_once` includes it only once.
`var_dump($value); print_r($array);`	Debug output: type and value, or a readable array.
`php -S localhost:8000`	Built-in development server from the current folder. Not for production.

Types and strict_types

`declare(strict_types=1);`	First statement in the file. Scalar type declarations are then enforced, not coerced.
`int, float, string, bool, array, null`	Basic types.
`?string`	Nullable: a string or null.
`int|float`	Union type.

`mixed, void, never, static, self`	Special types for parameters and return values.
`gettype($x) / get_debug_type($x)`	Type name; `get_debug_type` gives class names and cleaner output.
`is_int($x), is_string($x), is_array($x)`	Type checks.
`(int) '42'`	Type cast.
`intdiv(7, 2)`	Integer division: 3.
`$a <=> $b`	Spaceship operator: -1, 0 or 1. Handy in sort callbacks.
`$a === $b`	Strict comparison of type and value. Prefer it to `==`.

Strings

`$full = $first . ' ' . $last;`	Concatenation with the dot operator.
`"Total: {$order->total}"`	Curly braces for properties and array items inside double quotes.
`strlen($s) / mb_strlen($s)`	Length in bytes / in characters. Use `mb_` functions for Hindi and other multibyte text.
`str_contains($s, 'php')`	True if the substring is present (PHP 8.0+).
`str_starts_with($s, 'https') / str_ends_with($s, '.pdf')`	Checks the start or end.
`strtolower($s) / ucfirst($s)`	Change case.
`trim($s)`	Removes whitespace from both ends.
`str_replace('-', ' ', $s)`	Replace every occurrence.
`explode(',', $csv) / implode(', ', $list)`	String to array and back.
`sprintf('%05.2f', 3.14159)`	Formatted string: "03.14".
`number_format(1234567.891, 2)`	"1,234,567.89". (Indian lakh grouping needs `NumberFormatter` with the en_IN locale.)
`$html = <<<HTML ... HTML;`	Heredoc for multi-line strings. Nowdoc (`<<<'HTML'`) does not interpolate.

Arrays and array functions

`$list = ['a', 'b', 'c'];`	Indexed array.
`$user = ['name' => 'Asha', 'city' => 'Pune'];`	Associative array.
`$list[] = 'd';`	Append an item.
`in_array('b', $list, true)`	Value present? Pass `true` for strict comparison.
`array_key_exists('city', $user) / isset($user['city'])`	Key present? `isset` is false when the value is null.
`array_map(fn($n) => $n * 2, $nums)`	Transform every item.
`array_filter($nums, fn($n) => $n > 0)`	Keep matching items (keys are preserved).
`array_reduce($nums, fn($c, $n) => $c + $n, 0)`	Reduce to one value.

`sort($a); rsort($a); asort($a); ksort($a); usort($a, fn($x, $y) => $x <=> $y);`	Sort in place by value, reverse, value keeping keys, key, or a custom rule.
`array_merge($a, $b) / [...$a, ...$b]`	Join arrays.
`array_column($rows, 'email', 'id')`	Pull one column from a list of rows, keyed by id.
`foreach ($user as $key => $value) { }`	Loop over keys and values.

Control flow and match

`if ($a > $b) { } elseif ($a === $b) { } else { }`	Conditionals.
`$label = $age >= 18 ? 'adult' : 'minor';`	Ternary.
`$name = $_GET['name'] ?? 'Guest';`	Null coalescing: default when the key is missing or null.
`$config['lang'] ??= 'en';`	Assign only if not already set.
`for ($i = 0; $i < 10; $i++) { }`	Counted loop.
`while ($row = $stmt->fetch()) { }`	Loop while a condition is truthy.
`break; continue;`	Leave the loop / skip to the next iteration.
`match ($status) { 'paid' => 'Done', 'failed', 'expired' => 'Retry', default => 'Pending' };`	Strict comparison, returns a value, no fall-through. Throws `UnhandledMatchError` if nothing matches and there is no default.
`match (true) { $score >= 90 => 'A', $score >= 75 => 'B', default => 'C' };`	Match on conditions.

Functions

`function add(int $a, int $b): int { return $a + $b; }`	Typed parameters and return type.
`function greet(string $name = 'friend'): string { }`	Default parameter value.
`function total(int ...$prices): int { return array_sum($prices); }`	Variadic parameter.
`greet(name: 'Asha');`	Named arguments (PHP 8.0+): any order, skip optional ones.
`$double = fn(int $n): int => $n * 2;`	Arrow function; captures outer variables by value automatically.
`$add = function ($n) use ($tax) { return $n + $tax; };`	Anonymous function; `use` imports outer variables.
`$len = strlen(...);`	First-class callable syntax (PHP 8.1+).
`$city = $user?->getAddress()?->city;`	Nullsafe operator: null instead of an error when a link is null.
`function fail(string $m): never { throw new RuntimeException($m); }`	`never`: the function never returns.

Classes and OOP

`public function __construct(private string $name, public readonly int $id) {}`	Constructor promotion declares and assigns properties in one place.

`$p = new Product('Pen', 1);`	Create an object.
`public / protected / private`	Visibility: anywhere / class and children / this class only.
`class Admin extends User { }`	Inheritance; call the parent with `parent::method()`.
`interface Payable { public function pay(int $paise): bool; }`	A contract; `class Upi implements Payable`.
`abstract class Shape { abstract public function area(): float; }`	Cannot be instantiated; children implement abstract methods.
`trait HasSlug { } ... use HasSlug;`	Reusable methods copied into a class.
`enum Status: string { case Active = 'active'; case Dropped = 'dropped'; }`	Backed enum (PHP 8.1+). `Status::from('active')` and `tryFrom()`.
`readonly class Money { }`	Readonly class (PHP 8.2+): every property is readonly.
`const string CURRENCY = 'INR';`	Typed class constant (PHP 8.3+).
`#[\Override] public function area(): float { }`	PHP 8.3 attribute: error if the method does not actually override a parent method.
`public static function create(): static { return new static(); }`	Static factory, called as `Product::create()`.

```php
<?php
declare(strict_types=1);

enum Plan: string
{
    case Basic = 'basic';
    case Pro = 'pro';

    public function pricePaise(): int
    {
        return match ($this) {
            Plan::Basic => 49900,
            Plan::Pro   => 99900,
        };
    }
}

final class Subscription
{
    public function __construct(
        public readonly string $email,
        public readonly Plan $plan,
    ) {}
}

$sub = new Subscription('asha@example.com', Plan::from('pro'));
echo $sub->plan->pricePaise(); // 99900
```

Superglobals, forms and sessions

`$_GET['page']` / `$_POST['email']`	Query string and form body values. Always treat as untrusted input.

`$_SERVER['REQUEST_METHOD'] === 'POST'`	Check how the page was requested.
`$_FILES['cv']['tmp_name']`	Uploaded file; move it with `move_uploaded_file()` after validating type and size.
`filter_var($email, FILTER_VALIDATE_EMAIL)`	Validates an email; returns the value or false.
`filter_input(INPUT_GET, 'id', FILTER_VALIDATE_INT)`	Reads and validates in one step (null if missing, false if invalid).
`htmlspecialchars($text, ENT_QUOTES, 'UTF-8')`	Escapes output for HTML to prevent XSS.
`session_start();`	Starts or resumes a session. Call before any output.
`$_SESSION['user_id'] = $id;`	Stores data in the session.
`session_regenerate_id(true);`	New session ID after login, to prevent session fixation.
`$_SESSION = []; session_destroy();`	Logs the user out.
`setcookie('theme', 'dark', ['expires' => time() + 86400 * 30, 'path' => '/', 'secure' => true, 'httponly' => true, 'samesite' => 'Lax']);`	Sets a cookie with secure options.
`header('Location: /thank-you'); exit;`	Redirect after a successful POST. Always `exit` afterwards.

PDO and passwords

`$pdo = new PDO('mysql:host=localhost;dbname=shop;charset=utf8mb4', $user, $pass);`	Connect to MySQL. Keep credentials out of version control.
`[PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]`	Throw exceptions on errors (the default since PHP 8.0).
`[PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC]`	Return rows as associative arrays.
`$stmt = $pdo->prepare('SELECT * FROM users WHERE email = ?');`	Prepared statement with a positional placeholder.
`$stmt->execute([$email]); $user = $stmt->fetch();`	Run it with values; fetch one row (false if none).
`$stmt = $pdo->prepare('INSERT INTO users (name, email) VALUES (:name, :email)');`	Named placeholders.
`$stmt->execute(['name' => $name, 'email' => $email]);`	Values are sent separately from SQL, which prevents SQL injection.
`$stmt->fetchAll() / $stmt->fetchColumn() / $pdo->lastInsertId()`	All rows / first column of the next row / ID of the row just inserted.
`$pdo->beginTransaction(); ... $pdo->commit(); // or rollBack()`	Group several queries so they succeed or fail together.
`$hash = password_hash($password, PASSWORD_DEFAULT);`	Hashes a password with a random salt. Store the whole string (allow 255 characters).
`password_verify($password, $hash)`	Checks a login attempt against the stored hash.
`password_needs_rehash($hash, PASSWORD_DEFAULT)`	True when the hash should be upgraded after a successful login.

```php
<?php
declare(strict_types=1);

$pdo = new PDO('mysql:host=localhost;dbname=shop;charset=utf8mb4', 'app', getenv('DB_PASS') ?: '', [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
]);

$stmt = $pdo->prepare('SELECT id, password_hash FROM users WHERE email = ?');
$stmt->execute([$_POST['email'] ?? '']);
$user = $stmt->fetch();

if ($user && password_verify($_POST['password'] ?? '', $user['password_hash'])) {
    session_start();
    session_regenerate_id(true);
    $_SESSION['user_id'] = $user['id'];
    header('Location: /dashboard');
    exit;
}
echo 'Invalid email or password.';
```

Error handling

`try { } catch (PDOException $e) { } finally { }`	Catch a specific exception; `finally` always runs.
`catch (InvalidArgumentException | DomainException $e)`	Catch several types in one block.
`catch (Throwable)`	Catch anything, including `Error`; the variable is optional since PHP 8.0.
`throw new InvalidArgumentException('Price must be positive');`	Throw an exception.
`$id = $_GET['id'] ?? throw new RuntimeException('Missing id');`	`throw` as an expression (PHP 8.0+).
`class PaymentFailed extends RuntimeException { }`	Custom exception type.
`error_reporting(E_ALL);`	Report every error, warning and notice.
`ini_set('display_errors', '0'); ini_set('log_errors', '1');`	Production: log errors instead of showing them to visitors.
`error_log('Import failed: ' . $e->getMessage());`	Writes to the server error log.
`set_exception_handler(fn(Throwable $e) => error_log((string) $e));`	Last-resort handler for uncaught exceptions.
`json_validate($json)`	PHP 8.3: checks that a string is valid JSON without decoding it.

Online version, with the matching free guide: skillnewton.com/resources/cheat-sheets/php-cheat-sheet/
Learn this hands-on: PHP Web Development: Foundation & Practical Implementation – skillnewton.com/courses/php-foundation-course/