

PHP Cheat Sheet

Hello World

```
<?php
echo 'Hello, World!';
```

PHP Tags

| Tag | Description |
|-----------|---------------------------------|
| <?php | Standard opening tag |
| <?= \$foo | Short echo tag |
| <? | Short opening tag (discouraged) |
| ?> | Standard closing tag |

Variables

```
$greeting = 'Hello, World!';
echo $greeting; // Hello, World!
```

Constants

```
const CONSTANT = 'value';
define('RUNTIME_CONSTANT', CONSTANT);
```

```
echo CONSTANT; // value
echo RUNTIME_CONSTANT; // value
```

Strings

```
$name = 'World!';
echo 'Hello, $name!'; // Hello, $name!
echo "Hello, $name!"; // Hello, World!
echo "Hello, {$name}!"; // Hello, World!
```

```
echo <<<END
This is a multi-line string
in HEREDOC syntax (with interpolation).
END;
```

```
echo <<<'END'
This is a multi-line string
in NOWDOC syntax (without interpolation).
END;
```

Integers

| Example | Value |
|---------|------------------|
| 28 | 28 |
| 10_000 | 10000 |
| -28 | -28 |
| 012 | 10 (octal) |
| 0x0A | 10 (hexadecimal) |
| 0b1010 | 10 (binary) |

Floats

| Example | Value |
|---------|-----------------------------|
| 1.234 | 1.234 |
| -1.2 | -1.2 |
| 1.2e3 | 1200 (scientific notation) |
| 7E-3 | 0.007 (scientific notation) |

Arrays

```
$array = [1, 2, 3];
$array[] = 4;
$array[4] = 5;
```

Functions

```
function foo(int $a, int $b = 5): int
{
    return $a + $b;
}
foo(1, 2); // 3
foo(1); // 6
```

Named Parameters

```
function foo(string $a, string $b): string
{
    return $a . $b;
}
foo(b: 'World!', a: 'Hello, '); // Hello, World!
```

Anonymous Functions (Closures)

```
$y = 3;
$foo = function(int $x) use ($y): int {
    return $x + $y;
};
$foo(1); // 4
```

Arrow Functions

```
$y = 3;
$foo = fn(int $x): int => $x + $y;
$foo(1); // 4
```

Generators

```
function generate(): iterable
{
    yield 1;
    yield 2;
}
```

```
foreach (generate() as $value) {
    echo $value;
}
```

Comments

```
// This is a one line C++ style comment
# This is a one line shell-style comment
/* This is a
   multi-line comment */
```

```
/**
 * This is a PHPDoc docblock
 * @param string[] $bar
 * @return void
 */
function foo(array $bar): void
{}
```

Atomic / Built-in Types

| Type | Description |
|-----------------|-----------------------------------|
| null | NULL (no value) |
| bool | Boolean (true or false) |
| int | Integer |
| float | Floating point number |
| string | String |
| array | Array |
| object | Object |
| resource | Reference to an external resource |
| callable | Callback function |
| void | Function does not return a value |
| never (PHP 8.1) | Function never terminates |
| false | false |
| true (PHP 8.2) | true |

Composite Types & Type Aliases

| Type | Description |
|----------------------|--------------------------------|
| ?string | Nullable type: string or null |
| string bool | Union type: string or bool |
| Foo&Bar (PHP 8.1) | Intersection type: Foo and Bar |
| (A&B) null (PHP 8.2) | Disjunctive Normal Form (DNF) |
| iterable | array or Traversable |
| mixed | Any type |

If/Else

```
if ($a > $b) {
    echo "a is greater than b";
} elseif ($a == $b) {
    echo "a is equal to b";
} else {
    echo "a is less than b";
}
```

While

```
while ($i < 10) {
    echo $i++;
}
```

Do/While

```
do {
    echo $i++;
} while ($i < 10);
```

For

```
for ($i = 0; $i < 10; $i++) {
    echo $i;
}
```

Foreach

```
foreach ($array as $value) {
    echo $value;
}
```

```
foreach ($array as $key => $value) {
    echo "$key: $value";
}
```

Switch

```
switch ($i) {
    case 0:
    case 1:
        echo "i equals 0 or 1";
        break;
    default:
        echo "i is not equal to 0 or 1";
}
```

Match

```
$foo = match ($i) {
    0 => "i equals 0",
    1, 2 => "i equals 1 or 2",
    default => "i is not equal to 0, 1 or 2",
};
```

Enumerations (PHP 8.1)

```
enum Suit {
    case Hearts;
    case Diamonds;
    case Clubs;
    case Spades;
}

$suit = Suit::Hearts;
$suit->name; // Hearts
```

Backed Enumerations (PHP 8.1)

```
enum Suit: string {
    case Hearts = '♥';
    case Diamonds = '♦';
    case Clubs = '♣';
    case Spades = '♠';
}

$hearts = Suit::from('♥');
$hearts->value; // '♥'
```

Language Constructs

| Construct | Description |
|---------------------|--|
| echo \$string | Output one or more strings |
| print \$string | Output a string and return 1 |
| unset(\$var) | Destroy the specified variable(s) |
| isset(\$var) | Determine if a variable is set |
| empty(\$var) | Determine if a variable is empty |
| die() | Output a message and terminate |
| exit() | Output a message and terminate |
| include <file> | Include and evaluate a file or throw a warning if it fails |
| require <file> | Include and evaluate a file or throw an error if it fails |
| include_once <file> | Include and evaluate a file once only or throw a warning if it fails |
| require_once <file> | Include and evaluate a file once only or throw an error if it fails |

Object-Oriented Programming

```
interface FooInterface
{
    public function baz(): string;
}

class Foo extends Bar implements FooInterface
{
    private string $bar;

    public const string BAZ = 'Hello, ';

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

    public function baz(): string
    {
        return self::BAZ . $this->bar;
    }
}
```

```
$foo = new Foo("World!");
echo $foo->baz(); // Hello, World!
echo Foo::BAZ; // Hello,
```

Class Keywords

| Keyword | Description |
|--------------------|------------------------------|
| abstract | Cannot be instantiated |
| final | Cannot be extended |
| extends | Extends another class |
| implements | Implements an interface |
| readonly (PHP 8.2) | All properties are read-only |

Method keywords

| Keyword | Description |
|----------|---|
| static | Can be called statically, cannot access <code>\$this</code> |
| abstract | Must be implemented by subclasses |
| final | Subclasses cannot override |

Property Keywords

| Keyword | Description |
|--------------------|------------------------------------|
| static | Can be accessed statically |
| readonly (PHP 8.1) | Can only be set in the constructor |

Method/Property/Constant Visibility

| Keyword | Accessible from |
|-----------|----------------------------------|
| public | Anywhere |
| protected | The current class and subclasses |
| private | The current class only |

Constructor Property Promotion

```
class Foo
{
    public function __construct(private string $bar)
    {}
}
```

Property Hooks (PHP 8.4)

```
class Foo
{
    public string $name {
        // (string $value) can be omitted
        set (string $value) {
            $this->name = strtolower($value);
        }
        // Short (closure-style) syntax
        get => ucfirst($this->name);
    }
}
```

Asymmetric Visibility (PHP 8.4)

```
class Foo
{
    public private(set) string $bar = 'baz';
}
```

```
$foo = new Foo();
echo $foo->bar; // baz
$foo->bar = 'Foobar'; // Error
```

Calling Methods/Properties/Constants

| Syntax | Calls <code>foo()</code> on... |
|-------------------------------|---|
| <code>\$foo->foo()</code> | The object referenced by <code>\$foo</code> |
| <code>\$this->foo()</code> | The current object (<code>\$this</code>) |
| <code>Foo::foo()</code> | The class <code>Foo</code> |
| <code>self::foo()</code> | The current class |
| <code>parent::foo()</code> | The parent (extended) class |
| <code>static::foo()</code> | The called class |

Namespacing and Importing

```
namespace Foo\Bar;

use Foo\Baz as BazAlias;
use Foo\Baz\{Qux, Quux};
use function strlen;
use const PHP_EOL;
```

Exceptions

```
try {
    throw new Exception('Something went wrong');
} catch (Exception $e) {
    // Code that runs when an exception is thrown
} finally {
    // Code that will always run
}
```

Traits

```
trait FooTrait
{
    public function baz(): string { ... }
}

class Foo
{
    use FooTrait;
}
```

Attributes

```
#[Attribute(flags: Attribute::TARGET_CLASS)]
class MyClassAttribute
{}
```

Arithmetic Operators

| Operator | Description |
|----------|----------------|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus |
| ** | Exponentiation |

Bitwise Operators

| Operator | Description |
|----------|-----------------|
| & | And |
| | Or (inclusive) |
| ^ | Xor (exclusive) |
| ~ | Not |
| << | Shift left |
| >> | Shift right |

Assignment Operators

| Operator | Description |
|----------|--------------------------------|
| = | Assign |
| += | Add and assign |
| -= | Subtract and assign |
| *= | Multiply and assign |
| /= | Divide and assign |
| %= | Modulus and assign |
| **= | Exponent and assign |
| &= | Bitwise and and assign |
| = | Bitwise or and assign |
| ^= | Bitwise xor and assign |
| <<= | Bitwise shift left and assign |
| >>= | Bitwise shift right and assign |

Comparison Operators

| Operator | Description |
|----------|---|
| == | Equal (values are converted) |
| === | Identical (values and types match) |
| != | Not equal |
| <> | Not equal |
| !== | Not identical |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| <=> | Returns -1, 0, or 1 if the first value is less than, equal to, or greater than the second value |

Incrementing/Decrementing Operators

| Operator | Description |
|----------|---|
| ++\$a | Increments \$a by one, then returns \$a |
| \$a++ | Returns \$a, then increments \$a by one |
| --\$a | Decrements \$a by one, then returns \$a |
| \$a-- | Returns \$a, then decrements \$a by one |

Logical Operators

| Operator | Description |
|----------|--------------|
| and | And |
| or | Or |
| xor | Exclusive or |
| ! | Not |
| && | And |
| | Or |

String Operators

| Operator | Description |
|----------|------------------------|
| . | Concatenate |
| .= | Concatenate and assign |

Other Operators

| Operator | Description |
|-----------------|--|
| \$a ? \$b : \$c | Ternary operator: return \$b if \$a is true, otherwise return \$c |
| \$a ?: \$b | Short ternary: return \$a if \$a is true, otherwise return \$b |
| \$a ?? \$b | Null coalescing: return \$a if \$a is not null, otherwise return \$b |
| \$a ??= \$b | Null coalescing assignment: assign \$b to \$a if \$a is null |
| \$a?->b | Nullsafe: return \$a->b if \$a is not null, otherwise return null |
| \$a = &\$b | Assign \$b by reference to \$a |
| @ | Suppress errors in the following expression |
| instanceof | Returns true if the left operand is an instance of the right operand |

Command Line Interface (CLI)

| Command | Description |
|-------------------------------|---|
| php <file> | Parse and execute <file> |
| php -l <file> | Syntax check <file> |
| php -r <code> | Run PHP <code> without using script tags |
| php -a | Run an interactive shell |
| php -S <addr>:<port> | Start built-in web server |
| php -S <addr>:<port> -t <dir> | Start built-in web server and specify document root |
| php -m | Show loaded modules |
| php -i | Show configuration information |
| php -v | Show PHP version |
| php -h | Show help |

String Functions

| Function | Description |
|--|---|
| <code>strlen(\$string)</code> | Return length of <code>\$string</code> |
| <code>str_replace(\$search, \$replace, \$subject)</code> | Replace <code>\$search</code> with <code>\$replace</code> in <code>\$subject</code> |
| <code>strstr(\$haystack, \$needle)</code> | Return part of <code>\$haystack</code> after <code>\$needle</code> |
| <code>substr(\$string, \$start, \$length)</code> | Return part of <code>\$string</code> starting at <code>\$start</code> |
| <code>strtolower(\$string)</code> | Return <code>\$string</code> in lowercase |
| <code>strtoupper(\$string)</code> | Return <code>\$string</code> in uppercase |
| <code>trim(\$string)</code> | Return <code>\$string</code> with whitespace trimmed |
| <code>ltrim(\$string)</code> | Return <code>\$string</code> with left whitespace trimmed |
| <code>rtrim(\$string)</code> | Return <code>\$string</code> with right whitespace trimmed |
| <code>explode(\$delimiter, \$string)</code> | Split <code>\$string</code> into an array by <code>\$delimiter</code> |
| <code>implode(\$glue, \$array)</code> | Join <code>\$array</code> into a string with <code>\$glue</code> |
| <code>str_repeat(\$string, \$multiplier)</code> | Repeat <code>\$string</code> <code>\$multiplier</code> times |

Math Functions

| Function | Description |
|---------------------------------|--|
| <code>abs(\$num)</code> | Return absolute value of <code>\$num</code> |
| <code>round(\$num)</code> | Round <code>\$num</code> to the nearest integer |
| <code>ceil(\$num)</code> | Round <code>\$num</code> up |
| <code>floor(\$num)</code> | Round <code>\$num</code> down |
| <code>max(\$a, \$b)</code> | Return the greater of <code>\$a</code> and <code>\$b</code> |
| <code>min(\$a, \$b)</code> | Return the lesser of <code>\$a</code> and <code>\$b</code> |
| <code>pow(\$a, \$b)</code> | Return <code>\$a</code> raised to the power of <code>\$b</code> |
| <code>rand(\$min, \$max)</code> | Return a random number between <code>\$min</code> and <code>\$max</code> |
| <code>sqrt(\$num)</code> | Return square root of <code>\$num</code> |

Array Functions

| Function | Description |
|--|---|
| <code>count(\$array)</code> | Count elements in <code>\$array</code> |
| <code>sort(\$array)</code> | Sort <code>\$array</code> |
| <code>array_merge(\$array1, \$array2)</code> | Merge <code>\$array1</code> and <code>\$array2</code> |
| <code>array_map(\$callback, \$array)</code> | Apply <code>\$callback</code> to each element of <code>\$array</code> |
| <code>array_filter(\$array, \$callback)</code> | Return elements of <code>\$array</code> for which <code>\$callback</code> returns true |
| <code>array_find(\$array, \$callback)</code> (PHP 8.4) | Return first element of <code>\$array</code> for which <code>\$callback</code> returns true |
| <code>array_find_key(\$array, \$callback)</code> (PHP 8.4) | Return key of the first element for which <code>\$callback</code> returns true |
| <code>array_any(\$array, \$callback)</code> (PHP 8.4) | Return true if <code>\$callback</code> returns true for any element of <code>\$array</code> |
| <code>array_all(\$array, \$callback)</code> (PHP 8.4) | Return true if <code>\$callback</code> returns true for all elements of <code>\$array</code> |
| <code>array_reduce(\$array, \$callback, \$initial)</code> | Reduce <code>\$array</code> to a single value using <code>\$callback</code> starting with <code>\$initial</code> |
| <code>array_slice(\$array, \$offset, \$length)</code> | Return part of <code>\$array</code> starting at <code>\$offset</code> and continuing for <code>\$length</code> elements |
| <code>array_keys(\$array)</code> | Return an array of keys from <code>\$array</code> |
| <code>array_values(\$array)</code> | Return an array of values from <code>\$array</code> |
| <code>array_combine(\$keys, \$values)</code> | Return an array of key/value pairs from <code>\$keys</code> and <code>\$values</code> |
| <code>array_reverse(\$array)</code> | Return a reversed copy of <code>\$array</code> |
| <code>array_search(\$needle, \$haystack)</code> | Return the key of <code>\$needle</code> in <code>\$haystack</code> |
| <code>array_unique(\$array)</code> | Return a copy of <code>\$array</code> with duplicate values removed |
| <code>array_diff(\$array1, \$array2)</code> | Return elements of <code>\$array1</code> not in <code>\$array2</code> |
| <code>array_intersect(\$array1, \$array2)</code> | Return elements of <code>\$array1</code> also in <code>\$array2</code> |

Filesystem Functions

| Function | Description |
|--|--|
| <code>file_exists(\$filename)</code> | Return true if <code>\$filename</code> exists |
| <code>is_dir(\$filename)</code> | Return true if <code>\$filename</code> is a directory |
| <code>is_file(\$filename)</code> | Return true if <code>\$filename</code> is a regular file |
| <code>is_readable(\$filename)</code> | Return true if <code>\$filename</code> is readable |
| <code>is_writable(\$filename)</code> | Return true if <code>\$filename</code> is writable |
| <code>mkdir(\$pathname)</code> | Create directory named <code>\$pathname</code> |
| <code>rmdir(\$dirname)</code> | Remove directory named <code>\$dirname</code> |
| <code>unlink(\$filename)</code> | Remove file named <code>\$filename</code> |
| <code>file_get_contents(\$filename)</code> | Return contents of <code>\$filename</code> |
| <code>file_put_contents(\$filename, \$data)</code> | Write <code>\$data</code> to <code>\$filename</code> |

php.ini Directives

| Directive | Description |
|--|---|
| <code>date.timezone</code> | Set default timezone |
| <code>error_reporting</code> | Set error reporting level (e.g. E_ALL, E_ERROR) |
| <code>display_errors</code> | Whether to display errors (e.g. On or Off) |
| <code>error_log</code> | Set error log file (e.g. /var/log/php.log) |
| <code>xdebug.mode</code> | Mode (e.g. debug, develop, profile) |
| <code>xdebug.discover_client_host</code> | Enable Xdebug to discover client host automatically |

Enable Xdebug Step Debugging

`XDEBUG_MODE=debug XDEBUG_SESSION=1 php <file>`

Or for web applications using a browser extension: [Firefox Helper](#) [Chrome Helper](#)