PHP Cheat Sheet

Hello World

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

PHP Tags

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

Variables

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

Constants

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

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

Strings

```php
$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

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

Functions

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

Named Parameters

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

Anonymous Functions (Closures)

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

Arrow Functions

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

Generators

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

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

Comments

```php
// 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

```php
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 `$this`
`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

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

Property Hooks (PHP 8.4)

```php
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)

```php
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 `foo()` on...
`$foo->foo()`	The object referenced by `$foo`
`$this->foo()`	The current object (`$this`)
`Foo::foo()`	The class `Foo`
`self::foo()`	The current class
`parent::foo()`	The parent (extended) class
`static::foo()`	The called class

Namespacing and Importing

```php
namespace Foo\Bar;

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

Exceptions

```php
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

```php
trait FooTrait
{
    public function baz(): string { ... }
}
class Foo
{
    use FooTrait;
}
```

Attributes

```php
#[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
strlen($string)	Return length of $string
str_replace($search, $replace, $subject)	Replace $search with $replace in $subject
strstr($haystack, $needle)	Return part of $haystack after $needle
substr($string, $start, $length)	Return part of $string starting at $start
strtolower($string)	Return $string in lowercase
strtoupper($string)	Return $string in uppercase
trim($string)	Return $string with whitespace trimmed
ltrim($string)	Return $string with left whitespace trimmed
rtrim($string)	Return $string with right whitespace trimmed
explode($delimiter, $string)	Split $string into an array by $delimiter
implode($glue, $array)	Join $array into a string with $glue
str_repeat($string, $multiplier)	Repeat $string $multiplier times

Math Functions

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

Array Functions

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

Filesystem Functions

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

php.ini Directives

Directive	Description
date.timezone	Set default timezone
error_reporting	Set error reporting level (e.g. E_ALL, E_ERROR)
display_errors	Whether to display errors (e.g. On or Off)
error_log	Set error log file (e.g. /var/log/php.log)
xdebug.mode	Mode (e.g. debug, develop, profile)
xdebug.discover_client_host	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