PHP cheatsheet

This PHP cheat sheet provides a reference for quickly looking up the correct syntax for the code you use most frequently.

# Getting Started

hello.php

```php
<?php // begin with a PHP open tag.

echo "Hello World\n";
print("Hello quickref.me");

?>
```

PHP run command

```
$ php hello.php
```

Variables

```php
$boolean1 = true;
$boolean2 = True;

$int = 12;
$float = 3.1415926;
unset($float);  // Delete variable

$str1 = "How are you?";
$str2 = 'Fine, thanks';
```

See: Types

Strings

```php
$url = "quickref.me";
echo "I'm learning PHP at $url";

// Concatenate strings
echo "I'm learning PHP at " . $url;

$hello = "Hello, ";
$hello .= "World!";
echo $hello;   # => Hello, World!
```

See: Strings

Arrays

```php
$num = [1, 3, 5, 7, 9];
$num[5] = 11;
unset($num[2]);    // Delete variable
print_r($num);     # => 1 3 7 9 11
echo count($num);  # => 5
```

See: Arrays

Operators

```php
$x = 1;
$y = 2;

$sum = $x + $y;
echo $sum;   # => 3
```

See: Operators

Include

vars.php

```php
<?php // begin with a PHP open tag.
$fruit = 'apple';
echo "I was imported";
return 'Anything you like.';
?>
```

test.php

Functions

Comments

```php
function add($num1, $num2 = 1) {
    return $num1 + $num2;
}
echo add(10);    # => 11
echo add(10, 5); # => 15
```

See: Functions

```php
# This is a one line shell-style comment

// This is a one line c++ style comment

/* This is a multi line comment
   yet another line of comment */
```

```php
<?php
include 'vars.php';
echo $fruit . "\n";   # => apple

/* Same as include,
cause an error if cannot be included*/
require 'vars.php';

// Also works
include('vars.php');
require('vars.php');

// Include through HTTP
include 'http://x.com/file.php';

// Include and the return statement
$result = include 'vars.php';
echo $result;  # => Anything you like.
?>
```

Constants

```php
const MY_CONST = "hello";

echo MY_CONST;   # => hello

# => MY_CONST is: hello
echo 'MY_CONST is: ' . MY_CONST;
```

Classes

```php
class Student {
    public function __construct($name) {
        $this->name = $name;
    }
}
$alex = new Student("Alex");
```

See: Classes

# PHP Types

Boolean

```php
$boolean1 = true;
$boolean2 = TRUE;
$boolean3 = false;
$boolean4 = FALSE;

$boolean5 = (boolean) 1;   # => true
$boolean6 = (boolean) 0;   # => false
```

Boolean are case-insensitive

Integer

```php
$int1 = 28;    # => 28
$int2 = -32;   # => -32
$int3 = 012;   # => 10 (octal)
$int4 = 0x0F;  # => 15 (hex)
$int5 = 0b101; # => 5  (binary)

# => 2000100000 (decimal, PHP 7.4.0)
$int6 = 2_000_100_000;
```

See also: Integers

Strings

```php
echo 'this is a simple string';
```

See: Strings

Arrays

```php
$arr = array("hello", "world", "!");
```

See: Arrays

Float (Double)

Null

Iterables

```php
$float1 = 1.234;
$float2 = 1.2e7;
$float3 = 7E-10;

$float4 = 1_234.567;  // as of PHP 7.4.0
var_dump($float4);    // float(1234.567)

$float5 = 1 + "10.5";   # => 11.5
$float6 = 1 + "-1.3e3"; # => -1299
```

```php
$a = null;
$b = 'Hello php!';
echo $a ?? 'a is unset'; # => a is unset
echo $b ?? 'b is unset'; # => Hello php

$a = array();
$a == null    # => true
$a === null   # => false
is_null($a)   # => false
```

```php
function bar(): iterable {
    return [1, 2, 3];
}
function gen(): iterable {
    yield 1;
    yield 2;
    yield 3;
}
foreach (bar() as $value) {
    echo $value;   # => 123
}
```

# PHP Strings

String

```php
# => '$String'
$sgl_quotes = '$String';

# => 'This is a $String.'
$dbl_quotes = "This is a $sgl_quotes.";

# => a    tab character.
$escaped   = "a \t tab character.";

# => a slash and a t: \t
$unescaped = 'a slash and a t: \t';
```

Multi-line

```php
$str = "foo";

// Uninterpolated multi-liners
$nowdoc = <<<'END'
Multi line string
$str
END;

// Will do string interpolation
$heredoc = <<<END
Multi line
$str
END;
```

Manipulation

```php
$s = "Hello Phper";
echo strlen($s);       # => 11

echo substr($s, 0, 3); # => Hel
echo substr($s, 1);    # => ello Phper
echo substr($s, -4, 3);# => hpe

echo strtoupper($s);   # => HELLO PHPER
echo strtolower($s);   # => hello phper

echo strpos($s, "l");      # => 2
var_dump(strpos($s, "L")); # => false
```

See: String Functions

# PHP Arrays

Defining

Multi array

Multi type

```php
$a1 = ["hello", "world", "!"]
$a2 = array("hello", "world", "!");
$a3 = explode(",", "apple,pear,peach");
```

Mixed int and string keys

```php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
    100   => -100,
    -100  => 100,
);
var_dump($array);
```

Short array syntax

```php
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
```

```php
$multiArray = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
];

print_r($multiArray[0][0]) # => 1
print_r($multiArray[0][1]) # => 2
print_r($multiArray[0][2]) # => 3
```

manipulation

```php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56;      // Append
$arr["x"] = 42;   // Add with key
sort($arr);       // Sort
unset($arr[5]);   // Remove
unset($arr);      // Remove all
```

See: Array Functions

```php
$array = array(
    "foo" => "bar",
    42    => 24,
    "multi" => array(
         "dim" => array(
             "a" => "foo"
         )
    )
);

# => string(3) "bar"
var_dump($array["foo"]);

# => int(24)
var_dump($array[42]);

# =>  string(3) "foo"
var_dump($array["multi"]["dim"]["a"]);
```

Indexing iteration

```php
$array = array('a', 'b', 'c');
$count = count($array);

for ($i = 0; $i < $count; $i++) {
    echo "i:{$i}, v:{$array[$i]}\n";
}
```

Value iteration

```php
$colors = array('red', 'blue', 'green');

foreach ($colors as $color) {
    echo "Do you like $color?\n";
}
```

Key iteration

```php
$arr = ["foo" => "bar", "bar" => "foo"];

foreach ( $arr as $key => $value )
{
        echo "key: " . $key . "\n";
    echo "val: {$arr[$key]}\n";
}
```

Concatenate arrays

```php
$a = [1, 2];
$b = [3, 4];

// PHP 7.4 later
```

Into functions

```php
$array = [1, 2];

function foo(int $a, int $b) {
        echo $a; # => 1
        echo $b; # => 2
```

Splat Operator

```php
function foo($first, ...$other) {
        var_dump($first); # => a
        var_dump($other); # => ['b', 'c']
}
foo('a', 'b', 'c' /*, ...*/ );
```

```
# => [1, 2, 3, 4]
$result = [...$a, ...$b];
```

```
}
foo(...$array);
```

```
// or
function foo($first, string ...$other){}
```

# PHP Operators

Arithmetic

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo
`**`	Exponentiation

Assignment

`a += b`	Same as `a = a + b`
`a -= b`	Same as `a = a – b`
`a *= b`	Same as `a = a * b`
`a /= b`	Same as `a = a / b`
`a %= b`	Same as `a = a % b`

Comparison

`==`	Equal
`===`	Identical
`!=`	Not equal
`<>`	Not equal
`!==`	Not identical
`<`	Less than
`>`	Greater than
`<=`	Less than or equal
`>=`	Greater than or equal
`<=>`	Less than/equal/greater than

Logical

`and`	And
`or`	Or
`xor`	Exclusive or
`!`	Not
`&&`	And
`||`	Or

Arithmetic

```
// Arithmetic
$sum        = 1 + 1; // 2
$difference = 2 - 1; // 1
$product    = 2 * 2; // 4
$quotient   = 2 / 1; // 2

// Shorthand arithmetic
$num = 0;
$num += 1;       // Increment $num by 1
echo $num++;     // Prints 1 (increments after evaluation)
echo ++$num;     // Prints 3 (increments before evaluation)
$num /= $float;  // Divide and assign the quotient to $num
```

Bitwise

`&`	And
`|`	Or (inclusive or)
`^`	Xor (exclusive or)
`~`	Not
`<<`	Shift left
`>>`	Shift right

# PHP Conditionals

If elseif else

Switch

Ternary operator

```php
$a = 10;
$b = 20;

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

```php
$x = 0;
switch ($x) {
    case '0':
        print "it's zero";
        break;
    case 'two':
    case 'three':
        // do something
        break;
    default:
        // do something
}
```

```php
# => Does
print (false ? 'Not' : 'Does');

$x = false;
# => Does
print($x ?: 'Does');

$a = null;
$b = 'Does print';
# => a is unset
echo $a ?? 'a is unset';
# => print
echo $b ?? 'b is unset';
```

Match

```php
$statusCode = 500;
$message = match($statusCode) {
  200, 300 => null,
  400 => 'not found',
  500 => 'server error',
  default => 'known status code',
};
echo $message; # => server error
```

See: Match

Match expressions

```php
$age = 23;

$result = match (true) {
    $age >= 65 => 'senior',
    $age >= 25 => 'adult',
    $age >= 18 => 'young adult',
    default => 'kid',
};

echo $result; # => young adult
```

# PHP Loops

while

```php
$i = 1;
# => 12345
while ($i <= 5) {
    echo $i++;
}
```

do while

```php
$i = 1;
# => 12345
do {
    echo $i++;
} while ($i <= 5);
```

for i

```php
# => 12345
for ($i = 1; $i <= 5; $i++) {
    echo $i;
}
```

break

```
# => 123
for ($i = 1; $i <= 5; $i++) {
    if ($i === 4) {
        break;
    }
    echo $i;
}
```

continue

```
# => 1235
for ($i = 1; $i <= 5; $i++) {
    if ($i === 4) {
        continue;
    }
    echo $i;
}
```

foreach

```
$a = ['foo' => 1, 'bar' => 2];
# => 12
foreach ($a as $k) {
    echo $k;
}
```

See: Array iteration

# PHP Functions

Returning values

```
function square($x)
{
    return $x * $x;
}

echo square(4);  # => 16
```

Return types

```
// Basic return type declaration
function sum($a, $b): float {/*...*/}
function get_item(): string {/*...*/}

class C {}
// Returning an object
function getC(): C { return new C; }
```

Nullable return types

```
// Available in PHP 7.1
function nullOrString(int $v) : ?string
{
    return $v % 2 ? "odd" : null;
}
echo nullOrString(3);       # => odd
var_dump(nullOrString(4));  # => NULL
```

See: Nullable types

Void functions

```
// Available in PHP 7.1
function voidFunction(): void
{
        echo 'Hello';
        return;
}

voidFunction();  # => Hello
```

Variable functions

```
function bar($arg = '')
{
    echo "In bar(); arg: '$arg'.\n";
}

$func = 'bar';
$func('test'); # => In bar(); arg: test
```

Anonymous functions

```
$greet = function($name)
{
    printf("Hello %s\r\n", $name);
};

$greet('World'); # => Hello World
$greet('PHP');   # => Hello PHP
```

Recursive functions

Default parameters

Arrow Functions

```php
function recursion($x)
{
    if ($x < 5) {
        echo "$x";
        recursion($x + 1);
    }
}
recursion(1);  # => 1234
```

```php
function coffee($type = "cappuccino")
{
    return "Making a cup of $type.\n";
}
# => Making a cup of cappuccino.
echo coffee();
# => Making a cup of .
echo coffee(null);
# => Making a cup of espresso.
echo coffee("espresso");
```

```php
$y = 1;

$fn1 = fn($x) => $x + $y;

// equivalent to using $y by value:
$fn2 = function ($x) use ($y) {
    return $x + $y;
};
echo $fn1(5);   # => 6
echo $fn2(5);   # => 6
```

# PHP Classes

Constructor

```php
class Student {
    public function __construct($name) {
        $this->name = $name;
    }
    public function print() {
        echo "Name: " . $this->name;
    }
}
$alex = new Student("Alex");
$alex->print();    # => Name: Alex
```

Inheritance

```php
class ExtendClass extends SimpleClass
{
    // Redefine the parent method
    function displayVar()
    {
        echo "Extending class\n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
```

Classes variables

```php
class MyClass
{
    const MY_CONST       = 'value';
    static $staticVar    = 'static';

    // Visibility
    public static $var1  = 'pubs';

    // Class only
    private static $var2 = 'pris';

    // The class and subclasses
    protected static $var3 = 'pros';

    // The class and subclasses
    protected $var6      = 'pro';

    // The class only
    private $var7        = 'pri';
}
```

Magic Methods

```php
class MyClass
{
    // Object is treated as a String
    public function __toString()
    {
        return $property;
```

Interface

```php
interface Foo
{
    public function doSomething();
}
interface Bar
{
    }
    // opposite to __construct()
    public function __destruct()
    {
        print "Destroying";
    }
```

```
    public function doSomethingElse();
}
class Cls implements Foo, Bar
{
    public function doSomething() {}
    public function doSomethingElse() {}
```

Access statically

```
echo MyClass::MY_CONST;   # => value
echo MyClass::$staticVar; # => static
```

# Miscellaneous

Basic error handling

```
try {
    // Do something
} catch (Exception $e) {
    // Handle exception
} finally {
    echo "Always print!";
}
```

Exception in PHP 8.0

```
$nullableValue = null;

try {
        $value = $nullableValue ?? throw new InvalidArgumentException();
} catch (InvalidArgumentException) { // Variable is optional
    // Handle my exception
    echo "print me!";
}
```

Custom exception

```
class MyException extends Exception {
    // do something
}
```

Usage

```
try {
    $condition = true;
    if ($condition) {
        throw new MyException('bala');
    }
} catch (MyException $e) {
```

Nullsafe Operator

```
// As of PHP 8.0.0, this line:
$result = $repo?->getUser(5)?->name;

// Equivalent to the following code:
if (is_null($repo)) {
    $result = null;
} else {
    $user = $repository->getUser(5);
    if (is_null($user)) {
        $result = null;
    } else {
        $result = $user->name;
    }
}
```

Regular expressions

```
$str = "Visit Quickref.me";
echo preg_match("/qu/i", $str); # => 1
```

See: Regex in PHP

fopen() mode

r	Read
r+	Read and write, prepend
w	Write, truncate
w+	Read and write, truncate
a	Write, append

```
    // Handle my exception
```

See also: Nullsafe Operator

a+	Read and write, append

Runtime defined Constants

```
define("CURRENT_DATE", date('Y-m-d'));

// One possible representation
echo CURRENT_DATE;   # => 2021-01-05

# => CURRENT_DATE is: 2021-01-05
echo 'CURRENT_DATE is: ' . CURRENT_DATE;
```