PHP Cheat Sheet - by HTMLCheatSheet.com

Editor

```php
<?php
$i = "World!";
echo "Hello " . $i;

// This is a comment
$prime = array(2,3,5,7,11);
for( $i = 0; $i < 5; $i += 1) {
  echo "\n" . $prime[$i];
}

function add($a, $b) {
    return $a + $b;
}
echo "\n30 + 7 = " . add(30, 7);
?>
```

Loops

For Loop

```php
// For loop count to 10
for( $i = 0; $i < 10; $i += 1) {
    echo $i . "\n";
}
```

Foreach Loop

```php
// Declare an array
$arr = array("tesla", "bmw", "audi");
// Loop through the array elements
foreach ($arr as $element) {
    echo "$element ";
}
```

While Loop

```php
// Declare a number
$i = 10;
// Counting down to 0
while ($i >= 0) {
  echo $i . "\n";
  $i--;
}
```

Do-While Loop

```php
$i = 10;
// Counting back to 0
do {
    echo $i . "\n";
    $i--;
} while ($i >= 0);
```

Useful Links

PHP.net	PHP Compiler Online
MS Copilot	Stack Overflow
DEV .to	PHP The Right Way

Basics

Hello World!

```php
<?php
$i = "World!";
echo "Hello ".$i;
?>
```

Comments

```php
<?php
// One liner

# another one liner

/* This is
a multiline
comment
*/
?>
```

Defining Functions

```php
function sayHello() {
    echo "Hello!";
}
sayHello(); // Outputs: Hello!
```

Variables

```php
$i = 1;
$pi = 3.14;
$stringName = "value";
$names = array("John","Jane","Jack");
var_dump($names);
```

var_dump

Dumps information about a variable.

```php
$a = array(1, 2, array("a", "b", "c"));
var_dump($a);
```

Objects

```php
class foo {
   function do_foo() {
     echo "Doing foo.";
    }
 }
$bar = new foo;
$bar->do_foo();
```

Escaping characters

```php
echo "\n"; //New line
```

Line feed	\n
Carriage return	\r
Horizontal tab space	\t
Vertical tab	\v
Escape characters	\e
Form (page or section separator)	\f
Backslash	\\
Dollar sign	\$
Single quote	\'
Double quote	\"

Operators

Conditions

If statement

```
if (condition) {
// execute this if condition is true
}
```

If - Else

```
if (condition) {
  // do this if condition is true
} else {
  // do this if condition is false
}
```

If - Elseif - Else

```
if (condition) {
  // code if condition is true
} elseif (condition2) {
  // condition is false and condition2 is true
} else {
  // if none of the conditions are met
}
```

Switch Statement

```
switch (n) {
case a:
  //code to execute if n=a;
break;
case b:
  //code to execute if n=b;
break;
case c:
  //code to execute if n=c;
break;
  // more cases as needed
default:
  // if n is neither of the above;
}
```

Ternary

Variable = (Condition) ? (Statement1) : (Statement2);

```
$result = condition ? value1 : value2;
```

Is the same as:

```
if (condition) {
        $result = value1;
} else {
        $result = value2;
}
```

Functions

Parameters

```
function greet($name) {
    echo "Hello " . $name;
}
greet("John");
// Outputs: Hello John
```

Operators

Hover your mouse for explanation.

$x + $y	Addition
$x - $y	Subtraction
$x * $y	Multiplication
$x / $y	Division
$x % $y	Modulus
$x ** $y	Exponentiation

Assignment

x = y	x = y
x = x + y	x += y
x = x - y	x -= y
x = x * y	x *= y
x = x / y	x /= y
x = x % y	x %= y

Comparison

$x == $y	Equal
$x === $y	Identical
$x != $y	Not equal
$x <> $y	Not equal
$x !== $y	Not identical
$x > $y	Greater than
$x < $y	Less than
$x >= $y	Greater than or equal to
$x <= $y	Less than or equal to
=++$x	Pre-increment
$x++	Post-increment
--$x	Pre-decrement
$x--	Post-decrement

Logical

!$x		Not
$x and $y	**$x && $y**	And
$x or $y	**$x || $y**	Or
$x xor $y		Xor

String

$s1 . $s2	Concatenation
$s1 .= $s2	Concatenation assignment

Arrays

$x + $y	Union
$x == $y	Equality
$x === $y	Identity
$x != $y	Inequality
$x <> $y	Inequality
$x !== $y	Non-identity

Arrays

Indexed arrays

```
$colors = array("Red", "Green", "Blue");
echo $colors[0]; // Outputs: Red
```

Associative arrays

Default Parameters

```php
function greet($name = "Visitor") {
    echo "Hello, " . $name;
}
greet(); // Outputs: Hello, Visitor
```

Return Values

```php
function add($a, $b) {
    return $a + $b;
}
$result = add(3, 5);
// $result is 8
```

Variable Scope

```php
$number = 10;
function multiplyByTwo() {
    global $number;
    $number *= 2;
}
multiplyByTwo();
echo $number; // Outputs: 20
```

Anonymous Functions

```php
$greet = function($name) {
    echo "Hello, " . $name;
};
$greet("Alice"); // Outputs: Hello, Alice
```

Built-in Functions

Built-in functions for various tasks, such as string manipulation, array handling, file operations etc..

- **String**: strlen(), str_replace(), substr()
- **Array**: array_merge(), array_pop(), array_keys()
- **Math**: abs(), ceil(), floor()
- **Date**: date(), strtotime(), mktime()
- **File**: fopen(), fwrite(), fread()

PHP Forms ✍️

$_GET

It's used for retrieving data from URL parameters (query string).

```php
// URL: html6.com?x=1&y=2
$x = $_GET['x'];  // $x will be 1
```

$_POST

Used for retrieving data from form submissions via HTTP POST method.

```php
$username = $_POST['username'];
// Variable will contain the value entered in the
```

Sent through form.php

```php
<form method="post" action="form.php">
  <input type="text" name="username">
   <input type="submit">
</form>
```

$_REQUEST

A general-purpose array that combines $_GET, $_POST, and $_COOKIE.

Example for the same HTML as above:

```php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $username = $_REQUEST['username'];
```

```php
$age = array("Peter" => 35, "Ben" => 37, "Joe"
echo $age['Peter']; // Outputs: 35
```

Multidimensional arrays

```php
$students = array(
  "John" => array("Math" => 123, "Science" => 4
  "Jane" => array("Math" => 789, "Science" => 9
);
echo $students['John']['Math']; // Outputs: 123
```

Array Functions

Hover your mouse for explanation

array_change_key_case, **array_chunk**, **array_column**, **array_combine**, **array_count_values**, **array_diff**, **array_diff_assoc**, **array_diff_key**, **array_diff_uassoc**, **array_diff_ukey**, **array_fill**, **array_fill_keys**, **array_filte**, **array_flip**, **array_intersect**, **array_intersect_assoc**, **array_intersect_key**, **array_intersect_uassoc**, **array_intersect_ukey**, **array_key_exists**, **array_keys**, **array_map**, **array_merge**, **array_merge_recursive**, **array_multisort**, **array_pad**, **array_pop**, **array_produc**, **array_push**, **array_rand**, **array_reduce**, **array_replace**, **array_replace_recursive**, **array_reverse**, **array_searcl**, **array_shift**, **array_slice**, **array_splice**, **array_sum**, **array_udiff**, **array_udiff_assoc**, **array_udiff_uassoc**, **array_uintersect**, **array_uintersect_assoc**, **array_uintersect_uassoc**, **array_unique**, **array_unshif**, **array_values**, **array_walk**, **array_walk_recursive**, **arsc**, **asort**, **compact**, **count**, **current**, **each**, **end**, **extract**, **in_array**, **key**, **krsort**, **ksort**, **list**, **natcasesort**, **natsort**, **next**, **prev**, **range**, **reset**, **rsort**, **shuffle**, **sort**, **uasort**, **uksort**, **usort**

Predefined Variables 🌎

$GLOBALS

They can be accessed from any scope. Variables in the outermost scope are automatically global and can be use inside functions. To use a global variable within a functio you must either declare it with the global keyword or use $GLOBALS syntax.

```php
$a = 5;
$b = 10;
function addition() {
  $GLOBALS['sum'] = $GLOBALS['a'] + $GLOBALS['b
}
addition();
echo $sum;
```

Super Global Variables

Hover your mouse for explanation

$_SERVER, **$_SERVER['PHP_SELF']**, **$_SERVER['GATEWAY_INTERFACE']**, **$_SERVER['SERVER_ADDR']**, **$_SERVER['SERVER_NAME']**, **$_SERVER['SERVER_SOFTWARE']**, **$_SERVER['SERVER_PROTOCOL']**, **$_SERVER['REQUEST_METHOD']**, **$_SERVER['REQUEST_TIME']**, **$_SERVER['QUERY_STRING']**, **$_SERVER['HTTP_ACCEPT']**, **$_SERVER['HTTP_ACCEPT_CHARSET']**, **$_SERVER['HTTP_HOST']**, **$_SERVER['HTTP_REFERER']**, **$_SERVER['HTTPS']**, **$_SERVER['REMOTE_ADDR']**, **$_SERVER['REMOTE_HOST']**, **$_SERVER['REMOTE_PORT']**, **$_SERVER['SCRIPT_FILENAME']**, **$_SERVER['SERVER_ADMIN']**, **$_SERVER['SERVER_PORT']**, **$_SERVER['SERVER_SIGNATURE']**

Predefined Functions

```
    echo "Hello, " . htmlspecialchars($username);
}
```

Security 👮

Always sanitize and validate user input on the server-side

- **htmlspecialchars()**: This function converts special characters to HTML entities.
- **trim()**: This function removes whitespace (or other characters specified) from the beginning and end of a string.
- **stripslashes()**: This function removes backslashes (\) from a string.

```
// Assuming form submission
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Sanitize and trim input
    $input = htmlspecialchars($_POST['input_field
    $input = trim($input);
    // Handle magic_quotes_gpc scenario if needed
    if (get_magic_quotes_gpc()) {
        $input = stripslashes($input);
    }
    // $input is now safe to use for further proc
}
```

Database 🖧

Connecting to DB

Procedural:

```
$mysqli = mysqli_connect("localhost", "username",
if (!$mysqli) {
    die("Connection failed: " . mysqli_connect_er
}
```

Object-Oriented:

```
$mysqli = new mysqli("localhost", "username", "pa
if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_
}
```

Queries

Procedural:

```
$result = mysqli_query($mysqli, "SELECT * FROM ta
if ($result) {
    while ($row = mysqli_fetch_assoc($result)) {
        echo $row["column_name"];
    }
}
```

Object-Oriented:

```
$result = $mysqli->query("SELECT * FROM table_nam
if ($result) {
    while ($row = $result->fetch_assoc()) {
        echo $row["column_name"];
    }
}
```

Prepared Statements

Procedural:

```
$stmt = mysqli_prepare($mysqli, "SELECT * FROM ta
mysqli_stmt_bind_param($stmt, "s", $value);
mysqli_stmt_execute($stmt);
$result = mysqli_stmt_get_result($stmt);
while ($row = mysqli_fetch_assoc($result)) {
    echo $row["column_name"];
}
mysqli_stmt_close($stmt);
```

Object-Oriented:

```
$stmt = $mysqli->prepare("SELECT * FROM table_nam
$stmt->bind_param("s", $value);
$stmt->execute();
```

boolval

Returns the boolean value of a variable

input	boolval(input)
0	false
12	true
0.0	false
1.2	true
""	false
"hello"	true
"0"	false
"1"	true
[1, 2]	true
[]	false
new stdClass	true

isset

To check whether a variable is empty

```
$x = 0;
if (isset($x)) {
echo "x is set";
}
// True because $x is set
```

unset

Unsets variable

```
$x = "Hello world!";
echo "before unset: " . $x;
unset($x);
echo "after unset: " . $a;
// Throws warning for undefined variable
```

debug_zval_dump

Provides a detailed dump of a variable's reference count and type information.

```
$a = "Hello, World!";
$b = $a;
$c = &$a;
debug_zval_dump($a);
```

string(13) "Hello, World!" interned

empty

Check whether a variable is empty or not

input	empty(input)
""	true
0	true
php	false

floatval

Converts a given variable to a floating-point number

input	floatval(input)
12.3	12.3
"12.3abc"	12.3
true	1
"abc"	0

get_defined_vars

Returns the resource type of a given resource variable

```
$result = $stmt->get_result();
while ($row = $result->fetch_assoc()) {
    echo $row["column_name"];
}
$stmt->close();
```

Inserting Data

Procedural:

```
$sql = "INSERT INTO table_name (column1, column2)
if (mysqli_query($mysqli, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error
}
```

Object-Oriented:

```
$sql = "INSERT INTO table_name (column1, column2)
if ($mysqli->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $mysqli->err
}
```

Updating Data

Procedural:

```
$sql = "UPDATE table_name SET column1 = 'value' W
if (mysqli_query($mysqli, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error
}
```

Object-Oriented:

```
$sql = "UPDATE table_name SET column1 = 'value' W
if ($mysqli->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error: " . $sql . "<br>" . $mysqli->err
}
```

Deleting Data

Procedural:

```
$sql = "DELETE FROM table_name WHERE condition";
if (mysqli_query($mysqli, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error
}
```

Object-Oriented:

```
$sql = "DELETE FROM table_name WHERE condition";
if ($mysqli->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error: " . $sql . "<br>" . $mysqli->err
}
```

Closing the Connection

Procedural:

```
mysqli_close($mysqli);
```

Object-Oriented:

```
$mysqli->close();
```

Common Functions Overview

- **Connecting**: mysqli_connect, mysqli_connect_error, mysqli_close
- **Queries**: mysqli_query, mysqli_fetch_assoc, mysqli_num_rows

```
// prints: stream
$fp = fopen("file", "w");
echo get_resource_type($fp) . "\n";
// prints: curl
$c = curl_init ();
echo get_resource_type($c) . "\n";
```

get_resource_type

Used to obtain the type of a given resource

```
// Open a file handle
$file = fopen("example.txt", "r");
// Get the type of the resource
echo get_resource_type($file); // Outputs: "str
// Close the file handle
fclose($file);
```

gettype

Used to determine the type of a given variable

$variable	gettype($variable)
12	integer
12.3	double
"HTML Cheat Sheet"	string
array(1, 2, 3)	array
fopen("file.txt", "r")	resource

intval

Integer value of a variable

$variable	intval($variable)
12	12
12.3	12
"12.3"	12
"101010"	intval("101010", 2) => 42
"Hello"	0

is_array

To check whether a given variable is an array

Regular Expressions

Syntax

$exp = "/cheatsheet/i";

```
$email = "admin@htmlcheatsheet.com";
if (preg_match('/^\w+([\.-]?\w+)*@\w+([\.-]?\w+
    echo "Valid email address";
} else {
    echo "Invalid email";
}
```

RegEx Functions

preg_match() - Returns 1 if the pattern was found in the string and 0 if not

preg_match_all() - Returns the number of times the patt was found in the string, which may also be 0

preg_replace() - Returns a new string where matched patterns have been replaced with another string

Modifiers

i - Performs a case-insensitive search

m - Performs a multiline search (patterns that search for beginning or end of a string will match the beginning or e of each line)

u - Enables correct matching of UTF-8 encoded patterns

- **Prepared Statemen**: mysqli_prepare, mysqli_stmt_bind_param, mysqli_stmt_execute, mysqli_stmt_get_result, mysqli_stmt_close
- **Error Handling**: mysqli_error, mysqli_stmt_error

Common Errors !

Don't forget the semicolon!

- Mismatched brackets **()**
- Incorrect quotes: **" '**
- Undefined variables
- Case sensitivity **$Var** ≠ **$var**
- Incorrect function calls that does not exist or with incorrect parameters.

File Inclusion Errors

Using **include** or **require** with an incorrect file path. Use absolute paths or ensure relative paths are correct.

SQL Injection

Failing to sanitize user inputs can lead to **SQL injection attacks**. Use prepared statements and parameterized queries.

Error Handling

Use **try-catch** blocks and implementing proper error handling mechanisms.

Incorrect Array Usage

Check if array keys exist before accessing them.

Session Handling

Start sessions with **session_start()** and handle session variables correctly.

Output Buffering

Unintentional output before headers are sent can cause "headers already sent" errors. Use output buffering or ensure no output before **header()** calls.

Scope Issues

Variable scope misunderstandings, such as trying to access a variable outside its defined scope. Use **global** keyword or pass variables as function arguments.

Misconfigured php.ini

Incorrect settings in the **php.ini** file can lead to various issues. Ensure configuration settings are appropriate for your environment.

Deprecated Features

Using deprecated functions or features that may be removed in future PHP versions. Refer to the latest PHP documentation.

Incorrect Timezone Configuration

Set the default timezone using **date_default_timezone_set()**.

Missing or Incorrect Encoding

Ensure correct character encoding is used, especially with multibyte strings.

Memory Limit Issues

Increase memory limit in **php.ini** or optimize code to use less memory.

Best Practices to Avoid Common PHP Errors

Patterns

[abc] – Find one character from the options between the brackets

[^abc] – Find any character NOT between the brackets

[0-9] – Find one character from the range 0 to 9

Metacharacters

| - Find a match for any one of the patterns separated by as in: cat|dog|fish

. - Find just one instance of any character

^ - Finds a match as the beginning of a string as in: ^Hel

**$** - Finds a match at the end of the string as in: World$

\d - Find a digit

\s -Find a whitespace character

\b - Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b

\uxxxx - Find the Unicode character specified by the hexadecimal number xxxx

Quantifiers

n+ - Matches any string that contains at least one n

n* - Matches any string that contains zero or more occurrences of n

n? - Matches any string that contains zero or one occurrences of n

n{x} - Matches any string that contains a sequence of X

n{x,y} - Matches any string that contains a sequence of X Y n's

n{x,} - Matches any string that contains a sequence of at least X n's

Grouping

Use parentheses ( ) to apply quantifiers to entire patterns select parts of the pattern to be used as a match.

Examples

```
$password = "HTMLcheatSheet123!";
if (preg_match('/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@
    echo "Valid password";
} else {
```

PHP Filters

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Define filters
    $filters = array(
        'email' => FILTER_VALIDATE_EMAIL,
// Validate email address and optionally, more
    );
    // Apply filters to $_POST data
    $sanitized_inputs = filter_input_array(INPU
    // Check if email is valid
    if ($sanitized_inputs['email'] === false) {
        echo "Invalid email address!";
    } else {
        // Use the sanitized input
        $email = $sanitized_inputs['email'];
        echo "Valid email address: " . $email;
    }
}
```

Filter Functions

- **Use Error Reporting**: Enable error reporting during development using **error_reporting(E_ALL)** and **ini_set('display_errors', 1)**.
- **Code Reviews**: Regular code reviews can catch errors early.
- **Testing**: Write unit tests and integration tests to cover different code paths.

Date and Time

```
// Get current timestamp
$currentTimestamp = time();
// Format and display current date and time
echo "Current timestamp: " . $currentTimestamp .
// Format date using date() function
echo "Current date: " . date("Y-m-d") . "<br>";
echo "Current time: " . date("H:i:s") . "<br>";
// Format date and time with specific timezone
date_default_timezone_set('America/New_York');
echo "Current date and time in New York: " . date
```

Formatting

- **d** - Days from 01 to 31
- **j** - Days 1 to 31
- **D** - Mon through Sun
- **l** - Sunday through Saturday
- **N** - 1 (for Mon) through 7 (for Sat)
- **w** - 0 (for Sun) through 6 (for Sat)
- **m** - Months, 01 through 12
- **n** - Months, 1 through 12
- **F** - January through December
- **M** - Jan through Dec
- **Y** - Four digits year (e.g. 2018)
- **y** - Two digits year (e.g. 18)
- **L** - Defines whether it's a leap year (1 or 0)
- **a** - am and pm
- **A** - AM and PM
- **g** - Hours 1 through 12
- **h** - Hours 01 through 12
- **G** - Hours 0 through 23
- **H** - Hours 00 through 23
- **i** - Minutes 00 to 59
- **s** - Seconds 00 to 59

Functions

Hover your mouse for explanation:

checkdate, **date_add**, **date_create_from_format**, **date_create**, **date_date_set**, **date_default_timezone_get**, **date_default_timezone_set**, **date_diff**, **date_format**, **date_get_last_errors**, **date_interval_create_from_date_string**, **date_interval_format**, **date_isodate_set**, **date_modify**, **date_offset_get**, **date_parse_from_format**, **date_parse**, **date_sub**, **date_sun_info**, **date_sunrise**, **date_sunset**, **date_time_set**, **date_timestamp_get**, **date_timestamp_set**, **date_timezone_get**, **date_timezone_set**, **date**, **getdate**, **gettimeofday**, **gmdate**, **gmmktime**, **gmstrftime**, **idate**, **localtime**, **microtime**, **mktime**, **strftime**, **strptime**, **strtotime**, **time**, **timezone_abbreviations_list**, **timezone_identifiers_list**, **timezone_location_get**, **timezone_name_from_abbr**, **timezone_name_get**, **timezone_offset_get**, **timezone_open**, **timezone_transitions_get**, **timezone_version_get**

- **filter_has_var()** - To check if a variable of the specified type exists
- **filter_id()** - Returns the ID belonging to a named filter
- **filter_input()** - Retrieves a specified external varia by name and optionally filters it
- **filter_input_array()** - Pulls external variables and optionally filters them
- **filter_list()** - Returns a list of all supported filters
- **filter_var_array()** - Gets multiple variables and optionally filters them
- **filter_var()** - Filters a variable with a specified filt

Filter Constants

- **FILTER_VALIDATE_BOOLEAN** - Validates a boolean
- **FILTER_VALIDATE_EMAIL** - Certifies an e-mail address
- **FILTER_VALIDATE_FLOAT** - Confirms a float
- **FILTER_VALIDATE_INT** - Verifies an integer
- **FILTER_VALIDATE_IP** - Validates an IP address
- **FILTER_VALIDATE_REGEXP** - Confirms a regul expression
- **FILTER_VALIDATE_URL** - Validates a URL
- **FILTER_SANITIZE_EMAIL** - Removes all illegal characters from an e-mail address
- **FILTER_SANITIZE_ENCODED** - Removes/Encoc special characters
- **FILTER_SANITIZE_MAGIC_QUOTES** - Applies **addslashes()**
- **FILTER_SANITIZE_NUMBER_FLOAT** - Remove characters, except digits, +- and .,eE
- **FILTER_SANITIZE_NUMBER_INT** - Gets rid of a characters except digits and + –
- **FILTER_SANITIZE_SPECIAL_CHARS** - Remove special characters
- **FILTER_SANITIZE_FULL_SPECIAL_CHARS** - Converts special characters to HTML entities
- **FILTER_SANITIZE_STRING** - Removes tags/spe characters from a string, same as: **FILTER_SANITIZE_STRIPPED**
- **FILTER_SANITIZE_URL** - Rids all illegal characte from a URL
- **FILTER_UNSAFE_RAW** - Do nothing, optionally strip/encode special characters
- **FILTER_CALLBACK** - Call a user-defined functio to filter data