Beginner’s Essential

PHP CHEAT SHEET

Fast, flexible and pragmatic scripting language.
# Table of Contents

- **PHP Basics** 2
- **Variables and Constants** 2
- **PHP Arrays - Grouped Values** 7
- **PHP Strings** 13
- **PHP Operators** 21
- **Loops in PHP** 23
- **Conditional Statements** 23
- **Working with Forms in PHP** 24
- **Regular Expressions** 26
- **PHP Functions** 28
- **PHP Filters** 29
- **HTTP Functions in PHP** 31
- **Working with MySQL** 31
- **Date and Time** 36
- **PHP Errors** 40
PHP BASICS

Including PHP in a File
<?php // place PHP code here ?>

Writing Comments
//
Denotes comments that only span on one line
#
Another way of producing single-line comments
/**/
Everything between /* and */ is not executed, also works across several lines

Outputting Data
<?php echo "<h1>PHP Cheat Sheet</h1>"; ?>

Writing PHP Functions
function NameOfTheFunction() {
    //place PHP code here
}

VARIABLES AND CONSTANTS

Defining Variables
<?php
    $BlogPostTitle = "PHP Cheat Sheet";
?>
Types of Data

Integers
Integers are non-decimal numbers between -2,147,483,648 and 2,147,483,647. They must have at least one digit and no decimal point. Can be in decimal, hexadecimal or octal.

Floats
This is the name for numbers with a decimal point or in exponential form.

Strings
This simply means text, we will talk about it in detail further below.

Boolean values
Meaning true/false statements.

Arrays
Arrays are variables that store several values. We will talk about them in detail further below.

Objects
Objects store both data and information on how to process it.

Resources
These are references to functions and resources outside of PHP.

NULL
A variable that is NULL doesn’t have any value.

Variable Scope
```php
function myFunction() {
    global $a, $b;
    $b = $a - $b;
    $b = $a - $b;
}
```
Predefined Variables

$GLOBALS
Used to access global variables from anywhere inside a PHP script.

$_SERVER
Contains information about the locations of headers, paths and scripts.

$_GET
Can collect data that was sent in the URL or submitted in an HTML form.

$_POST
Used to gather data from an HTML form and to pass variables.

$_REQUEST
Also collects data after submitting an HTML form

Variable-handling Functions

boolval
Used to retrieve the boolean value of a variable

debug_zval_dump
Outputs a string representation of an internal zend value

empty
Checks whether a variable is empty or not

floatval
Get the float value of a variable (doubleval is another possibility)

get_defined_vars
Returns an array of all defined variables

get_resource_type
Returns the resource type

gettype
Retrieves the variable type
import_request_variables
Import GET/POST/Cookie variables into the global scope

intval
Find the integer value of a variable

is_array
Checks whether a variable is an array

is_bool
Finds out if a variable is a boolean

is_callable
Verify whether you can call the contents of a variable as a function

is_countable
Check whether the contents of a variable are countable

is_float
Find out if the type of a variable is float, alternatives: is_double and is_real

is_int
Check if the type of a variable is an integer, is_integer and is_long also works

is_iterable
Verify that a variable’s content is an iterable value

is_null
Checks whether a variable’s value is NULL

is_numeric
Find out if a variable is a number or a numeric string

is_object
Determines whether a variable is an object

is_resource
Check if a variable is a resource

is_scalar
Tests if a variable is a scalar
**is_string**
Find out whether the type of a variable is a string

**isset**
Determine if a variable has been set and is not NULL

**print_r**
Provides human-readable information about a variable

**serialize**
Generates a representation of a value that is storable

**settype**
Sets a variable’s type

**strval**
Retrieves the string value of a variable

**unserialize**
Creates a PHP value from a stored representation

**unset**
Unsets a variable

**var_dump**
Dumps information about a variable

**var_export**
Outputs or returns a string representation of a variable that can be parsed

---

**Constants**

`define(name, value, true/false)`

Aside from user-defined constants, there also a number of default PHP constants:

`__LINE__`
Denotes the number of the current line in a file

`__FILE__`
Is the full path and filename of the file
### PHP ARRAYS – GROUPED VALUES

#### Indexed arrays
Arrays that have a numeric index

#### Associative arrays
Arrays where the keys are named

#### Multidimensional arrays
Arrays that contain one or more other arrays

### Declaring an Array in PHP
```php
<?php
    $cms = array("WordPress", "Joomla", "Drupal");
?
```

### Array Functions

**array_change_key_case**
Changes all keys in an array to uppercase or lowercase
array_chunk
Splits an array into chunks

array_column
Retrieves the values from a single column in an array

array_combine
Merges the keys from one array and the values from another into a new array

array_count_values
Counts all values in an array

array_diff
Compares arrays, returns the difference (values only)

array_diff_assoc
Compares arrays, returns the difference (values and keys)

array_diff_key
Compares arrays, returns the difference (keys only)

array_diff_uassoc
Compares arrays (keys and values) through a user callback function

array_diff_ukey
Compares arrays (keys only) through a user callback function

array_fill
Fills an array with values

array_fill_keys
Fills an array with values, specifying keys

array_filter
Filters the elements of an array via a callback function

array_flp
Exchanges all keys in an array with their associated values

array_intersect
Compare arrays and return their matches (values only)
array_intersect_assoc
Compare arrays and return their matches (keys and values)

array_intersect_key
Compare arrays and return their matches (keys only)

array_intersect_uassoc
Compare arrays via a user-defined callback function (keys and values)

array_intersect_ukey
Compare arrays via a user-defined callback function (keys only)

array_key_exists
Checks if a specified key exists in an array, alternative: key_exists

array_keys
Returns all keys or a subset of keys in an array

array_map
Applies a callback to the elements of a given array

array_merge
Merge one or several arrays

array_merge_recursive
Merge one or more arrays recursively

array_multisort
Sorts multiple or multi-dimensional arrays

array_pad
Inserts a specified number of items (with a specified value) into an array

array_pop
Deletes an element from the end of an array

array_product
Calculate the product of all values in an array

array_push
Push one or several elements to the end of the array
array_rand
Pick one or more random entries out of an array

array_reduce
Reduce the array to a single string using a user-defined function

array_replace
Replaces elements in the first array with values from following arrays

array_replace_recursive
Recursively replaces elements from later arrays into the first array

array_reverse
Returns an array in reverse order

array_search
Searches the array for a given value and returns the first key if successful

array_shift
Shifts an element from the beginning of an array

array_slice
Extracts a slice of an array

array_splice
Removes a portion of the array and replaces it

array_sum
Calculate the sum of the values in an array

array_udiff
Compare arrays and return the difference using a user function (values only)

array_udiff_assoc
Compare arrays and return the difference using a default and a user function (keys and values)

array_udiff_uassoc
Compare arrays and return the difference using two user functions (values and keys)

array_uintersect
Compare arrays and return the matches via user function (values only)

array_uintersect_assoc

Compare arrays and return the matches via a default user function (keys and values)

array_uintersect_uassoc

Compare arrays and return the matches via two user functions (keys and values)

array_unique

Removes duplicate values from an array

array_unshift

 Adds one or more elements to the beginning of an array

array_values

Returns all values of an array

array_walk

Applies a user function to every element in an array

array_walk_recursive

Recursively applies a user function to every element of an array

arsort

Sorts an associative array in descending order according to the value

asort

Sorts an associative array in ascending order according to the value

compact

Create an array containing variables and their values

count

Count all elements in an array, alternatively use sizeof

current

Returns the current element in an array, an alternative is pos

each

Return the current key and value pair from an array

del
Set the internal pointer to the last element of an array

```
extract
```

Import variables from an array into the current symbol table

```
in_array
```

Checks if a value exists in an array

```
key
```

Fetches a key from an array

```
krsort
```

Sorts an associative array by key in reverse order

```
ksort
```

Sorts an associative array by key

```
list
```

Assigns variables as if they were an array

```
natcasesort
```

Sorts an array using a “natural order” algorithm independent of case

```
natsort
```

Sorts an array using a “natural order” algorithm

```
next
```

Advance the internal pointer of an array

```
prev
```

Move the internal array pointer backwards

```
range
```

Creates an array from a range of elements

```
reset
```

Set the internal array pointer to its first element

```
rsort
```

Sort an array in reverse order

```
shuffle
```

Shuffle an array

```
sort
```

WebsiteSetup.org - Beginner’s PHP Cheat Sheet
Sorts an indexed array in ascending order

uasort

Sorts an array with a user-defined comparison function

uksort

Arrange an array by keys using a user-defined comparison function

usort

Categorize an array by values using a comparison function defined by the user

**PHP STRINGS**

**Defining Strings**

**Single quotes**

This is the simplest way. Just wrap your text in ' markers and PHP will handle it as a string.

**Double quotes**

As an alternative you can use ".. When you do, it’s possible to use the escape characters below to display special characters.

**heredoc**

Begin a string with <<< and an identifier, then put the string in a new line. Close it in another line by repeating the identifier. heredoc behaves like double-quoted strings.

**nowdoc**

Is what heredoc is for double-quoted strings but for single quotes. It works the same way and eliminates the need for escape characters.

**Escape Characters**

\n — Line feed
\r — Carriage return
\t — Horizontal tab
\v — Vertical tab
\e — Escape
\f — Form feed
\\ — Backslash
\$ — Dollar sign
String Functions

addcslashes()
Returns a string with backslashes in front of specified characters

addslashes()
Returns a string with backslashes in front of characters that need to be escaped

bin2hex()
Converts a string of ASCII characters to hexadecimal values

chop()
Removes space or other characters from the right end of a string

chr()
Returns a character from a specified ASCII value

chunk_split()
Splits a string into a series of smaller chunks

convert_cyr_string()
Converts a string from a Cyrillic character set to another

convert_uudecode()
Decodes a uuencoded string

convert_uuencode()
Encodes a string using uuencode

count_chars()
Returns information about the characters in a string

crc32()
Calculates a 32-bit CRC for a string

crypt()
Returns a hashed string

`echo()` or `echo ''`

Outputs one or several strings

`explode()`

Breaks down a string into an array

`fprintf()`

Writes a formatted string to a specified output stream

`get_html_translation_table()`

Returns the translation table used by `htmlspecialchars()` and `htmlentities()`

`hebrew()`

Transforms Hebrew text to visual

`texthebrew()`

Converts Hebrew text to visual text and implements HTML line breaks

`hex2bin()`

Translate hexadecimal values to ASCII characters

`html_entity_decode()`

Turns HTML entities to characters

`htmlentities()`

Converts characters to HTML entities

`htmlspecialchars_decode()`

Transforms special HTML entities to characters

`htmlspecialchars()`

Switches predefined characters to HTML entities

`implode()`

Retrieves a string from the elements of an array, same as `join()`

`lcfirst()`

Changes a string’s first character to lowercase

`levenstein()`
Calculates the Levenshtein distance between two strings

`localeconv()`
Returns information about numeric and monetary formatting for the locale

`ltrim()`
Removes spaces or other characters from the left side of a string

`md5()`
Calculates the MD5 hash of a string and returns it

`md5_file()`
Calculates the MD5 hash of a file

`metaphone()`
Provides the metaphone key of a string

`money_format()`
Returns a string as a currency string

`nl_langinfo()`
Gives specific locale information

`nl2br()`
Inserts HTML line breaks for each new line in a string

`number_format()`
Formats a number including grouped thousands

`ord()`
Returns the ASCII value of a string’s first character

`parse_str()`
Parses a string into variables

`print()`
Outputs one or several strings

`printf()`
Outputs a formatted string

`quoted_printable_decode()`
Converts a quoted-printable string to 8-bit binary

`quoted_printable_encode()`

Goes from 8-bit string to a quoted-printable string

`quotemeta()`

Returns a string with a backslash before metacharacters

`rtrim()`

Strips whitespace or other characters from the right side of a string

`setlocale()`

Sets locale information

`sha1()`

Calculates a string’s SHA-1 hash

`sha1_file()`

Does the same for a file

`similar_text()`

Determines the similarity between two strings

`sounex()`

Calculates the soundex key of a string

`sprintf()`

Returns a formatted string

`sscanf()`

Parses input from a string according to a specified format

`str_getcsv()`

Parses a CSV string into an array

`str_ireplace()`

Replaces specified characters in a string with specified replacements (case-insensitive)

`str_pad()`

Pads a string to a specified length

`str_repeat()`
Repeats a string a preset number of times

`str_replace()`
Replaces specified characters in a string (case-sensitive)

`str_rot13()`
Performs ROT13 encoding on a string

`str_shuffle()`
Randomly shuffles the characters in a string

`str_split()`
Splits strings into arrays

`str_word_count()`
Returns the number of words in a string

`strcasecmp()`
Case-insensitive comparison of two strings

`strcmp()`
Binary safe string comparison (case sensitive)

`strcoll()`
Compares two strings based on locale

`strcspn()`
Returns the number of characters found in a string before the occurrence of specified characters

`strip_tags()`
Removes HTML and PHP tags from a string

`stripcslashes()`
Opposite of `addcslashes()`

`stripslashes()`
Opposite of `addslashes()`

`stripos()`
Finds the position of the first occurrence of a substring within a string (case insensitive)

`stristr()`
Case-insensitive version of strstr()

strlen()
Returns the length of a string

strnatcasecmp()
Case-insensitive comparison of two strings using a "natural order" algorithm

strnatcmp()
Same as the aforementioned but case sensitive

strncasecmp()
String comparison of a defined number of characters (case insensitive)

strncmp()
Same as above but case-sensitive

strpbrk()
Searches a string for any number of characters

strpos()
Returns the position of the first occurrence of a substring in a string (case sensitive)

strrchr()
Finds the last occurrence of a string within another string

strrev()
Reverses a string

strripos()
Finds the position of the last occurrence of a string’s substring (case insensitive)

strrpos()
Same as strripos() but case sensitive

strposn()
The number of characters in a string with only characters from a specified list

strstr()
Case-sensitive search for the first occurrence of a string inside another string

`strtok()`
Splits a string into smaller chunks

`strtolower()`
Converts all characters in a string to lowercase

`strtoupper()`
Same but for uppercase letters

`strtr()`
Translates certain characters in a string, alternative: `strchr()`

`substr()`
Returns a specified part of a string

`substr_compare()`
Compares two strings from a specified start position up to a certain length, optionally case sensitive

`substr_count()`
Counts the number of times a substring occurs within a string

`substr_replace()`
Replaces a substring with something else

`trim()`
Removes space or other characters from both sides of a string

`ucfirst()`
Transforms the first character of a string to uppercase

`ucwords()`
Converts the first character of every word in a string to uppercase

`vfprintf()`
Writes a formatted string to a specified output stream

`vprintf()`
Outputs a formatted string

`vsprintf()`
Writes a formatted string to a variable

```
wordwrap()
```

Shortens a string to a given number of characters

**PHP OPERATORS**

**Arithmetic Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+</code></td>
<td>Addition</td>
</tr>
<tr>
<td><code>-</code></td>
<td>Subtraction</td>
</tr>
<tr>
<td><code>*</code></td>
<td>Multiplication</td>
</tr>
<tr>
<td><code>/</code></td>
<td>Division</td>
</tr>
<tr>
<td><code>%</code></td>
<td>Modulo (the remainder of value divided by another)</td>
</tr>
<tr>
<td><code>**</code></td>
<td>Exponentiation</td>
</tr>
</tbody>
</table>

**Assignment Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>+=</code></td>
<td>a += b is the same as a = a + b</td>
</tr>
<tr>
<td><code>-=</code></td>
<td>a -= b is the same as a = a - b</td>
</tr>
<tr>
<td><code>*=</code></td>
<td>a *= b is the same as a = a * b</td>
</tr>
<tr>
<td><code>/=</code></td>
<td>a /= b is the same as a = a / b</td>
</tr>
<tr>
<td><code>%=</code></td>
<td>a %= b is the same as a = a % b</td>
</tr>
</tbody>
</table>

**Comparison Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>Equal</td>
</tr>
<tr>
<td><code>===</code></td>
<td>Identical</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>Not equal</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Not equal</td>
</tr>
<tr>
<td><code>!==</code></td>
<td>Not identical</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>Less than</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>Greater than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>Less than or equal to</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td><code>&lt;&gt;</code></td>
<td>Less than, equal to, or greater than</td>
</tr>
</tbody>
</table>

**Logical Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>and</code></td>
<td>And</td>
</tr>
<tr>
<td><code>or</code></td>
<td>Or</td>
</tr>
<tr>
<td><code>xor</code></td>
<td>Exclusive or</td>
</tr>
<tr>
<td><code>!</code></td>
<td>Not</td>
</tr>
<tr>
<td><code>&amp;&amp;</code></td>
<td>And</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
</tbody>
</table>
Bitwise Operators

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

Error Control Operator

You can use the @ sign to prevent expressions from generating error messages. This is often important for security reasons, for example to keep confidential information safe.

Execution Operator

PHP supports one execution operator, which is ``` (backticks). These are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command.

Increment/Decrement Operators

```$v```
Increments a variable by one, then returns it

$v++
Returns a variable, then increments it by one

``$v``
Decrements the variable by one, returns it afterward

$v--
Returns the variable then decrements it by one

String Operators

.
Used to concatenate (mean combine) arguments

.=
Used to append the argument on the right to the left-side argument
LOOPS IN PHP

For Loop
for (starting counter value; ending counter value; increment by which
   to increase) {
  // code to execute goes here
}

Foreach Loop
foreach ($InsertYourArrayName as $value) {
  // code to execute goes here
}

While Loop
while (condition that must apply) {
  // code to execute goes here
}

Do..While Loop
do {
  // code to execute goes here;
} while (condition that must apply);

CONDITIONAL STATEMENTS

If Statement
if (condition) {
  // code to execute if condition is met
}

If..Else
if (condition) {
  // code to execute if condition is met
} else {
  // code to execute if condition is not met
}
If..Elseif..Else

```php
if (condition) {
    // code to execute if condition is met
} elseif (condition) {
    // code to execute if this condition is met
} else {
    // code to execute if none of the conditions are met
}
```

Switch Statement
```
switch (n) {
    case x:
        code to execute if n=x;
        break;
    case y:
        code to execute if n=y;
        break;
    case z:
        code to execute if n=z;
        break;
    // add more cases as needed
    default:
        code to execute if n is neither of the above;
}
```

WORKING WITH FORMS IN PHP

Using GET vs POST

GET collects data via URL parameters. That means all variable names and their values are contained in the page address.

The advantage of this is that you’re able to bookmark the information. Keep in mind that it also means that the information is visible to everyone. For that reason, GET is not suitable for sensitive information such as passwords. It also limits the amount of data that can be sent in ca 2000 characters.

POST, on the other hand, uses the HTTP POST method to pass on variables. This makes the data invisible to third parties, as it is sent in the HTTP body. You are not able to bookmark it.
With POST, there are no limits to the amount of information you can send. Aside from that, it also has advanced functionality and is therefore preferred by developers.

**Form Security**

PHP offers tools to thwart those attacks, namely:

- htmlspecialchars()
- trim()
- stripslashes()

**Required Fields, Error Messages and Data Validation**

Aside from that, PHP is able to define required fields (you can’t submit the form without filling them out), display error messages if some information is missing and to validate data. We have already talked about the necessary tools to do so.

For example, you can simply define variables for your form fields and use the empty() function to check if they have values. After that, create a simple if/else statement to either send the submitted data or output an error message.

The next step is to check submitted data for validity. For that, PHP offers a number of filters such as FILTER_VALIDATE_EMAIL to make sure a submitted email address has the right format.

**Regular Expressions (RegEx)**

**Syntax**

```
$exp = "/w3schools/i";
```

**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 pattern 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

RegEx Modifiers
i
Performs a case-insensitive search
m
Performs a multiline search (patterns that search for the beginning or end of a string will match the beginning or end of each line)
u
Enables correct matching of UTF-8 encoded patterns

RegEx 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: ^Hello
$
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's

n{x,y}
Matches any string that contains a sequence of X to 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. They can also be used to select parts of the pattern to be used as a match.

```php
<?php
    $str = "Apples and bananas.";
    $pattern = "/ba(na){2}/i";
    echo preg_match($pattern, $str); // Outputs 1
?>
```
PHP Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Syntax

function functionName() {
    code to be executed;
}
functionName();

Function Arguments

```php
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

Default Argument Value

```php
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Returning values

```php
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```
PHP FILTERS

Filter Functions

filter_has_var()
Checks if a variable of the specified type exists

filter_id()
Returns the ID belonging to a named filter

filter_input()
Retrieves a specified external variable 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 filter

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 regular expression

FILTER_VALIDATE_URL
Validates a URL

FILTER_SANITIZE_EMAIL
Removes all illegal characters from an e-mail address

FILTER_SANITIZE_ENCODED
Removes/Encodes special characters

FILTER_SANITIZE_MAGIC_QUOTES
Applies addslashes()

FILTER_SANITIZE_NUMBER_FLOAT
Removes all characters, except digits, +- and .eE

FILTER_SANITIZE_NUMBER_INT
Gets rid of all characters except digits and + –

FILTER_SANITIZE_SPECIAL_CHARS
Removes special characters

FILTER_SANITIZE_FULL_SPECIAL_CHARS
Converts special characters to HTML entities

FILTER_SANITIZE_STRING
Removes tags/special characters from a string, alternative:
FILTER_SANITIZE_STRIPPED

FILTER_SANITIZE_URL
Rids all illegal characters from a URL

FILTER_UNSAFE_RAW
Do nothing, optionally strip/encode special characters

FILTER_CALLBACK
Call a user-defined function to filter data
HTTP FUNCTIONS IN PHP

HTTP Functions

header()
Sends a raw HTTP header to the browser

headers_list()
A list of response headers ready to send (or already sent)

headers_sent()
Checks if and where the HTTP headers have been sent

setcookie()
Defines a cookie to be sent along with the rest of the HTTP headers

setrawcookie()
Defines a cookie (without URL encoding) to be sent along

WORKING WITH MYSQL

MySQL Functions

mysqli_affected_rows()
The number of affected rows in the previous MySQL operation

mysqli_autocommit()
Turn auto-committing database modifications on or off

mysqli_change_user()
Changes the user of the specified database connection

mysqli_character_set_name()
The default character set for the database connection

mysqli_close()
Closes an open database connection

mysqli_commit()
Commits the current transaction
mysqli_connect_errno()
The error code from the last connection error

mysqli_connect_error()
The error description from the last connection error

mysqli_connect()
Opens a new connection to the MySQL server

mysqli_data_seek()
Moves the result pointer to an arbitrary row in the result set

mysqli_debug()
Performs debugging operations

mysqli_dump_debug_info()
Dumps debugging information into a log

mysqli_errno()
The last error code for the most recent function call

mysqli_error_list()
A list of errors for the most recent function call

mysqli_error()
The last error description for the most recent function call

mysqli_fetch_all()
Fetches all result rows as an array

mysqli_fetch_array()
Fetches a result row as an associative, a numeric array, or both

mysqli_fetch_assoc()
Fetches a result row as an associative array

mysqli_fetch_field_direct()
Metadata for a single field as an object

mysqli_fetch_field()
The next field in the result set as an object
mysqli_fetch_fields()
An array of objects that represent the fields in a result set

mysqli_fetch_lengths()
The lengths of the columns of the current row in the result set

mysqli_fetch_object()
The current row of a result set as an object

mysqli_fetch_row()
 Fetches one row from a result set and returns it as an enumerated array

mysqli_field_count()
The number of columns for the most recent query

mysqli_field_seek()
Sets the field cursor to the given field offset

mysqli_field_tell()
The position of the field cursor

mysqli_free_result()
Frees the memory associated with a result

mysqli_get_charset()
A character set object

mysqli_get_client_info()
The MySQL client library version

mysqli_get_client_stats()
Returns client per-process statistics

mysqli_get_client_version()
The MySQL client library version as an integer

mysqli_get_connection_stats()
Statistics about the client connection

mysqli_get_host_info()
The MySQL server hostname and the connection type
mysqli_get_proto_info()
The MySQL protocol version

mysqli_get_server_info()
Returns the MySQL server version

mysqli_get_server_version()
The MySQL server version as an integer

mysqli_info()
Returns information about the most recently executed query

mysqli_init()
Initializes MySQLi and returns a resource for use with mysqli_real_connect()

mysqli_insert_id()
Returns the auto-generated ID used in the last query

mysqli_kill()
Asks the server to kill a MySQL thread

mysqli_more_results()
Checks if there are more results from a multi query

mysqli_multi_query()
Performs one or more queries on the database

mysqli_next_result()
Prepares the next result set from mysqli_multi_query()

mysqli_num_fields()
The number of fields in a result set

mysqli_num_rows()
The number of rows in a result set

mysqli_options()
Sets extra connect options and affect behavior for a connection

mysqli_ping()
Pings a server connection or tries to reconnect if it has gone down
mysqli_prepare()
Prepares an SQL statement for execution

mysqli_query()
Performs a query against the database

mysqli_real_connect()
Opens a new connection to the MySQL server

mysqli_real_escape_string()
Escapes special characters in a string for use in an SQL statement

mysqli_real_query()
Executes an SQL query

mysqli_reap_async_query()
Returns the result from async query

mysqli_refresh()
Refreshes tables or caches or resets the replication server information

mysqli_rollback()
Rolls back the current transaction for the database

mysqli_select_db()
Changes the default database for the connection

mysqli_set_charset()
Sets the default client character set

mysqli_set_local_infile_default()
Unsets a user-defined handler for the LOAD LOCAL INFILE command

mysqli_set_local_infile_handler()
Sets a callback function for the LOAD DATA LOCAL INFILE command

mysqli_sqlstate()
Returns the SQLSTATE error code for the last MySQL operation

mysqli_ssl_set()
Establishes secure connections using SSL
mysqli_stat()
The current system status

mysqli_stmt_init()
Initializes a statement and returns an object for use with
mysqli_stmt_prepare()

mysqli_store_result()
Transfers a result set from the last query

mysqli_thread_id()
The thread ID for the current connection

mysqli_thread_safe()
Returns if the client library is compiled as thread-safe

mysqli_use_result()
Initiates the retrieval of a result set from the last query executed
using the mysqli_real_query()

mysqli_warning_count()
The number of warnings from the last query in the connection

**DATE AND TIME**

Date/Time Functions

checkdate()
Checks the validity of a Gregorian date

date_add()
Adds a number of days, months, years, hours, minutes and seconds to a
date object

date_create_from_format()
Returns a formatted DateTime object

date_create()
Creates a new DateTime object

date_date_set()
Sets a new date
date_default_timezone_get()
Returns the default timezone used by all functions

date_default_timezone_set()
Sets the default timezone

date_diff()
Calculates the difference between two dates

date_format()
Returns a date formatted according to a specific format

date_get_last_errors()
Returns warnings or errors found in a date string

date_interval_create_from_date_string()
Sets up a DateInterval from relative parts of a

string date_interval_format()
Formats an interval

date_isodate_set()
Sets a date according to ISO 8601 standards

date_modify()
Modifies the timestamp

date_offset_get()
Returns the offset of the timezone

date_parse_from_format()
Returns an array with detailed information about a specified date,
according to a specified format

date_parse()
Returns an array with detailed information about a specified date

date_sub()
Subtracts days, months, years, hours, minutes and seconds from a date

date_sun_info()
Returns an array containing information about sunset/sunrise and
twilight begin/end for a specified day and location
date_sunrise()
The sunrise time for a specified day and location

date_sunset()
The sunset time for a specified day and location

date_time_set()
Sets the time

date_timestamp_get()
Returns the Unix timestamp

date_timestamp_set()
Sets the date and time based on a Unix timestamp

date_timezone_get()
Returns the time zone of a given DateTime object

date_timezone_set()
Sets the time zone for a DateTime object

date()
Formats a local date and time

gdate()
Date/time information of a timestamp or the current local date/time

gmtime()
The current time

gmdate()
Formats a GMT/UTC date and time

gmmktime()
The Unix timestamp for a GMT date

gmstrftime()
Formats a GMT/UTC date and time according to locale settings

idate()
Formats a local time/date as an integer
localtime()
The local time

microtime()
The current Unix timestamp with microseconds

mktime()
The Unix timestamp for a date

strftime()
Formats a local time and/or date according to locale settings

strptime()
Parses a time/date generated with strftime()

strtotime()
Transforms an English textual DateTime into a Unix timestamp

time()
The current time as a Unix timestamp

timezone_abbreviations_list()
Returns an array containing dst, offset, and the timezone name

timezone_identifiers_list()
An indexed array with all timezone identifiers

timezone_location_get()
Location information for a specified timezone

timezone_name_from_abbr()
Returns the timezone name from an abbreviation

timezone_name_get()
The name of the timezone

timezone_offset_get()
The timezone offset from GMT

timezone_open()
Creates a new DateTimeZone object
timezone_transitions_get()
Returns all transitions for the timezone

timezone_version_get()
Returns the version of the timezonedb

Date and Time Formatting

- d — 01 to 31
- j — 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)
- I — 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

PHP ERRORS

Error Functions

d debug_backtrace()
Used to generate a backtrace

d debug_print_backtrace()
Prints a backtrace

er error_get_last()
Gets the last error that occurred
error_log()
Sends an error message to the web server’s log, a file or a mail account

error_reporting()
Specifies which PHP errors are reported

restore_error_handler()
Reverts to the previous error handler function

restore_exception_handler()
Goes back to the previous exception handler

set_error_handler()
Sets a user-defined function to handle script errors

set_exception_handler()
Sets an exception handler function defined by the

user trigger_error()
Generates a user-level error message, you can also use user_error()

Error Constants

E_ERROR
Fatal run-time errors that cause the halting of the script and can’t be recovered from

E_WARNING
Non-fatal run-time errors, execution of the script continues

E_PARSE
Compile-time parse errors, should only be generated by the parser

E_NOTICE
Run-time notices that indicate a possible error

E_CORE_ERROR
Fatal errors at PHP initialization, like an E_ERROR in PHP core

E_CORE_WARNING
Non-fatal errors at PHP startup, similar to E_WARNING but in PHP core
E_COMPILE_ERROR
Fatal compile-time errors generated by the Zend Scripting Engine

E_COMPILE_WARNING
Non-fatal compile-time errors by the Zend Scripting Engine

E_USER_ERROR
Fatal user-generated error, set by the programmer using trigger_error()

E_USER_WARNING
Non-fatal user-generated warning

E_USER_NOTICE
User-generated notice by trigger_error()

E_STRICT
Suggestions by PHP to improve your code (needs to be enabled)

E_RECOVERABLE_ERROR
Catchable fatal error caught by a user-defined handle

E_DEPRECATED
Enable this to receive warnings about a code which is not future-proof

E_USER_DEPRECATED
User-generated warning for deprecated code

E_ALL
All errors and warnings except E_STRICT