Table of Contents

Variable Handling Functions

Variables in PHP are containers for storing data.

- A variable name is preceded with a $ sign.
- It must start with a variable or an underscore.
- A variable cannot be started with a number.
- A variable can only contain alphanumeric characters.
- Variables are case-sensitive. $Result and $result are treated as two different variables.

Variable Functions allow us to define, use and manipulate variables as required.

1.	boolval	Returns the boolean value of a variable.
2.	debug_zval_dump	Dumps the string representation of an internal zend value to output.
3.	doubleval	Alias of floatval.
4.	empty	Determines whether a variable is empty.
5.	floatval	Returns the Float Value of a variable.
6.	get_defined_vars	Returns an Array of all defined variable.
7.	get_resource_type	Returns the resource type.
8.	gettype	Returns the type of a variable.
9.	import_request_variables	Imports GET/POST/Cookie variables into the resource type.
10.	intval	Get the integer value of the variable.
11.	is_array	Determines whether the variable is an array
12.	is_bool	Determines whether a variable is Boolean

13.	is_callable	Verifies if the contents of a variable can be called a function.
14.	is_float | is double | is_real	Determines whether the type of a variable is ‘float’.
15.	is_int| is_integer | is_long	Determines whether the type of variable is an integer.
16.	is_null	Determines whether a variable is NULL
17.	is_numeric	Determines whether a variable is a number or a numeric string.
18.	is_object	Determines whether a variable is an object.
19.	is_resource	Determines whether a variable is a resource.
20.	is_scalar	Determines whether a variable is scalar.
21.	is_string	Determines whether the type of variable is string.
22.	isset	Determines if a variable is set and not NULL.
23.	print_r	Prints information in human readable format.
24.	serialize	Generates a storable representation of a value.
25.	settype	Set the type of a variable.
26.	strval	Get the string value of a variable.
27.	unserialize	Creates a PHP value of a stored variable.
28.	Unset	Unsets a given variable
29.	var_dump	Dumps information about a variable.
30.	var_export	Outputs or returns a parsable string representation of a variable.

Array Functions

An array is used to store a list of values in one single variable. The values can be accessed by referring to an index number which always starts with 0. Arrays can be:

- **Indexed:** Default Array Format. Values can be accessed by index number.
- **Associative:** Use named keys.
- **Multidimensional:** Can also be referred as a matrix. It contains one or more arrays. You can also think of it as a nested array.

Array Functions allow us to define, store, traverse data and information as required.

1.	array()	Declares and creates an Array.
2.	array_change_key_case()	Returns an array with all keys in lowercase or uppercase.
3.	array_chunk()	Splits an array into chunks of Array.
4.	array_combine()	Creates an array by using one array for keys and one for values.
5.	array_count_values()	Returns an array with the number of occurrences for each value.
6.	array_diff()	Compares array values and returns the differences.
7.	array_diff_assoc()	Compares array key and values and returns the differences.
8.	array_diff_key()	Compares array keys and returns the differences.
9.	array_diff_uassoc()	Compares array keys and values, with an additional user made function check, and returns the differences.
10.	array_diff_ukey()	Compares array keys, with an additional user made function check, and returns the differences.
11.	array_fill()	Fills an array with values.
12.	array_filter()	Filters elements of an array through a user-made function.
13.	array_flip()	Exchanges all keys with their associated values in the array.
14.	array_intersect()	Compares array values and returns the matches
15.	array_intersect_assoc()	Compares array keys and values and returns the matches.
16.	array_intersect_key()	Compares array keys and returns the matches.
17.	array_intersect_uassoc()	Compares keys and values, with an additional user-made function check and return the matches.
18.	array_intersect_ukey()	Compares the keys, with an additional user-made function check and return the matches.
19.	array_key_exists()	Determines if the specified keys exists in the array.
20.	array_keys()	Returns all keys of an array.
21.	array_map()	Send each value of an array to a user-made function, which returns new values.
22.	array_merge() | array_merge_recursive()	Merges one or more arrays into one array.
23.	array_multisort()	Sorts multiple or multi-dimensional arrays.

24.	array_pad()	Inserts a specified number of items, with a specified value, to an array.
25.	array_pop()	Deletes the last element of an array.
26.	array_product()	Calculates the product of the values in an array.
27.	array_push()	Inserts one or more elements to the end of an array.
28.	array_rand()	Returns one or more random keys from an array.
29.	array_reduce()	Returns an array as a string, using a user defined function.
30.	array_reverse()	Returns an array in the reverse order.
31.	array_search()	Searches an array for a given value and returns the key.
32.	array_shift()	Removes the first element from an array, and returns the value of the removed element.
33.	array_slice()	Removes and replaces specified elements of an array.
34.	array_splice()	Removes and replaces specified elements of an array
35.	array_sum()	Returns the sum of the values in an array.
36.	array_udiff()	Compares array values in a user-made function and returns an array.
37.	array_udiff_assoc()	Compares array keys, and compares array values in a user-made function, and returns an array.
38.	array_udiff_uassoc()	Compares array keys and array values in user-made functions, and returns an array.
39.	array_uintersect()	Compares array values in a user-made function and returns an array.
40.	array_uintersect_assoc()	Compares array keys, and compares array values in a user-made function, and returns an array.
41.	array_uintersect_uassoc()	Compares array keys and array values is user-made functions, and returns an array.
42.	array_unique()	Removes duplicate values from an array.
43.	array_unshift()	Adds one or more elements to the beginning of an array.
44.	array_values()	Returns all the values of an array.
45.	array_walk()	Applies a user function to every member of an array.
46.	array_walk_recursive()	Applies a user function recursively to every member of an array.
47.	arsort()	Sorts an array in reverse order and maintains index association.
48.	asort()	Sorts an array and maintains index association.
49.	compact()	Create array containing variables and their values.

50.	count()	Counts elements in an array, or properties in an object.
51.	current()	Returns the current element in an array
52.	each()	Returns the current key and value pair from an array
53.	end()	Sets the internal pointer of an array to its last element.
54.	extract()	Imports variables into the current symbol table from an array.
55.	in_array()	Checks if a specified value exists in an array.
56.	key()	Fetches a key from an array.
57.	krsort()	Sorts an array by key in reverse order.
58.	ksort()	Sorts an array by key.
59.	list()	Assigns variables as if they were an array.
60.	natcasesort()	Sorts an array using a case insensitive.
61.	natsort()	Sorts an array
62.	next()	Advance the internal array pointer of an array
63.	pos()	Alias of current.
64.	prev()	Rewinds the internal array pointer.
65.	range()	Creates an array containing a range of elements.
66.	reset()	Sets the internal pointer of an array to its first element.
67.	rsort()	Sorts an array in reverse order.
68.	shuffle()	Shuffles an array
69.	sizeof()	Alias of count()
70.	sort()	Sorts an array
71.	uasort()	Sorts an array with a user-defined function and maintain index association.
72.	uksort()	Sorts an array by keys using a user-defined function
73.	usort()	Sorts an array by values using a user-defined function.

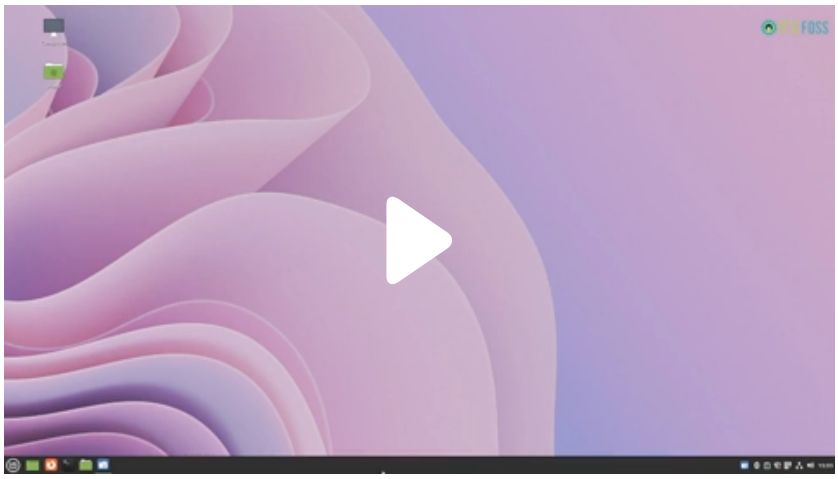
Array Constants

1.	CASE_LOWER	Used with array_change_key_case() to convert array keys to lowercase

2.	CASE_UPPER	Used with array_change_key_case() to convert array keys to uppercase
3.	SORT_ASC	Used with array_multisort() to sort in ascending order
4.	SORT_DESC	Used with array_multisort() to sort in descending order
5.	SORT_REGULAR	Used to compare items normally
6.	SORT_NUMERIC	Used to compare items numerically
7.	SORT_STRING	Used to compare items as strings
8.	SORT_LOCALE_STRING	Used to compare items as strings, based on the current locale
9.	COUNT_NORMAL	Counts all elements in an array.
10.	COUNT_RECURSIVE	Recursively counts the elements in an array.
11.	EXTR_OVERWRITE	Imports variables from an array into the current symbol table.
12.	EXTR_SKIP	It is an extraction flag. If there is a collision, do not overwrite the existing variable.
13.	EXTR_PREFIX_SAME	If there is a collision, prefix the variable name with **prefix**.
14.	EXTR_PREFIX_ALL	Prefix all variable names with **prefix**.
15.	EXTR_PREFIX_INVALID	Only prefix invalid/numeric variable names with **prefix**.
16.	EXTR_PREFIX_IF_EXISTS	Only create prefixed variable names if the non-prefixed version of the same variable exists in the current symbol table.
17.	EXTR_IF_EXISTS	Only overwrite the variable if it already exists in the current symbol table, otherwise do nothing. This is useful for defining a list of valid variables and then extracting only those variables defined out of $_REQUEST.
18.	EXTR_REFS	Extracts variables as references. This means the values of the imported variables are still referencing the values of the array parameter.

String Functions

Strings are a sequence of characters. It is a data type. It can be alphanumeric as well. Strings are created by declaring a variable and assigning string characters to it. String Functions allow us to manipulate and handle strings as required.



1.	addcslashes()	Returns a string with backslashes in front of the specified characters.
2.	addslashes()	Returns a string with backslashes in front of predefined characters.
3.	bin2hex()	Converts a string of ASCII characters to hexadecimal values.
4.	chop()	Alias of rtrim()
5.	chr()	Returns a character from a specified ASCII value.
6.	chunk_split()	Split a string into a series of smaller parts.
7.	convert_cyr_string()	Converts a string from one Cyrillic character-set to another.
8.	convert_uudecode()	Decodes a uuencoded string.
9.	convert_uuencode()	Encodes a string using the uuencode algorithm.
10.	count_chars()	Returns how many times an ASCII character occurs within a string and returns the information.
11.	crc32()	Calculates a 32-bit CRC for a string.
12.	crypt()	One-way string encryption (hashing)

13.	echo()	Outputs strings
14.	explode()	Breaks a string into an array.
15.	fprintf()	Writes a formatted string to a specified output stream.
16.	get_html_translation_table()	Returns the translation table used by htmlspecialchars() and htmlentities()
17.	hebrev()	Converts Hebrew text to visual text.
18.	hebrevc()	Converts Hebrew text to visual text and new lines(\n) into
19.	html_entity_decode()	Converts HTML entities to characters.
20.	htmlentities()	Converts characters to HTML entities.
21.	htmlspecialchars_decode()	Converts some predefined HTML entities to characters.
22.	htmlspecialchars()	Converts some predefined characters to HTML entities.
23.	implode()	Returns a string from the elements of an array.
24.	join()	Alias of implode()
25.	levenshtein()	Returns the Levenshtein distance between two strings.
26.	localeconv()	Returns locale numeric and monetary formatting information.
27.	ltrim()	Strips whitespaces from the left side of a string.
28.	md5()	Calculates the md5 hash of a string.
29.	md5_file()	Calculates the md5 hash of a file.
30.	metaphone()	Calculates the metaphone key of a string.
31.	money_format()	Returns a string formatted as a currency string.
32.	nl_langinfo()	Returns a specific local information.
33.	nl2br()	Inserts HTML line breaks in front of each newline in a string.
34.	number_format()	Formats a number with grouped thousands.
35.	ord()	Returns the ASCII value of the first character of a string.
36.	parse_str()	Parses a query string into variables.
37.	print()	Outputs a string.
38.	printf()	Outputs a formatted string.

39.	quoted_printable_decode()	Decodes a quoted-printable string.
40.	quotemeta()	Quotes meta-characters.
41.	rtrim()	Strips whitespace from the right side of a string.
42.	setlocale()	Sets locale information.
43.	sha1()	Calculates the SHA-1 hash of a string.
44.	sha1_file()	Calculates the SHA-1 hash of a file.
45.	similar_text()	Calculates the similarity between two strings.
46.	soundex()	Calculates the soundex key of a string
47.	sprintf()	Writes a formatted string to a variable.
48.	sscanf()	Parses input from a string according to a format.
49.	str_ireplace()	Replaces some characters in a string(case-insensitive)
50.	str_pad()	Pads a string to a new length.
51.	str_repeat()	Repeats a string a specified number of times.
52.	str_replace()	Replaces some characters in a string(case-sensitive)
53.	str_rot13()	Performs the ROT13 encoding on a string.
54.	str_shuffle()	Randomly shuffles all characters in a string.
55.	str_split()	Splits a string into an array.
56.	str_word_count()	Count the number of words in a string.
57.	strcasecmp()	Compares two strings (case-insensitive)
58.	strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
59.	strcmp()	Compares two strings (case-sensitive)

60.	strcoll()	Locale based string comparison
61.	strcspn()	Returns the number of characters found in a string before any part of some specified characters are found.
62.	strip_tags()	Strips HTML and PHP tags from a string.
63.	stripcslashes()	Unquotes a string quoted with addcslashes()
64.	stripslashes()	Unquotes a string quoted with addslashes()
65.	stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
66.	stristr()	Finds the first occurrence of a string inside another string(case-insensitive)
67.	strlen()	Returns the length of a string
68.	strnatcasecmp()	Compares two strings
69.	strnatcmp()	Compares two strings
70.	strncasecmp()	String comparison of the first n characters (case - insensitive)
71.	strncmp()	String comparison of the first n characters (case – sensitive)
72.	strpbrk()	Searches a string for any of a set of characters.
73.	strpos()	Returns the position of the first occurrence of a string inside another string(case-sensitive)
74.	strrchr()	Finds the last occurrence of a string inside another string.
75.	strrev()	Reverses a string
76.	strripos()	Finds the position of the last occurrence of a string inside another string(case-insensitive)
77.	strrpos()	Finds the position of the last occurrence of a string inside another string (case – sensitive)
78.	strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist.
79.	strstr()	Finds the first occurrence of a string inside another string(case-sensitive)
78.	strtok()	Splits a string into smaller strings.
79.	strtolower()	Converts a string to lowercase letters.
80.	strtoupper()	Converts a string to uppercase letters.

81.	strtr()	Translates certain characters in a string.
82.	substr()	Returns a part of a string.
83.	substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
84.	substr_count()	Counts the number of times a substring occurs in a string.
85.	substr_replace()	Replaces a part of a string with another string.
86.	trim()	Strips whitespace from both sides of a string.
87.	ucfirst()	Converts the first character of a string to uppercase.
88.	ucwords()	Converts the first character of each word in a string to uppercase.
89.	vfprintf()	Writes a formatted string to a specified output stream.
90.	vprintf()	Outputs a formatted string.
91.	vsprintf()	Writes a formatted string to a variable.
92.	wordwrap()	Wraps a string to a given number of characters.

String Constants

1.	CRYPT_SALT_LENGTH	Contains the length of the default encryption method for the system. For standard DES encryption, the length is 2
2.	CRYPT_STD_DES	Set to 1 if the standard DES-based encryption with a 2 character salt is supported, 0 otherwise
3.	CRYPT_EXT_DES	Set to 1 if the extended DES-based encryption with a 9 character salt is supported, 0 otherwise
4.	CRYPT_MD5	Set to 1 if the MD5 encryption with a 12 character salt starting with 1 is supported, 0 otherwise
5.	CRYPT_BLOWFISH	Set to 1 if the Blowfish encryption with a 16 character salt starting with 2 or $2a$ is supported, 0 otherwise
6.	HTML_SPECIALCHARS	Converts some predefined characters to HTML entities.
7.	HTML_ENTITIES	Converts a special character in an input string into the form of an HTML character.
8.	ENT_COMPAT	Converts double quotes
9.	ENT_QUOTES	Will convert both single and double quotes.
10.	ENT_NOQUOTES	Will leave both single and double quotes unconverted.
11.	ENT_IGNORE	Silently discard invalid code unit sequences instead of returning an empty string. Using this flag is discouraged as it may have security implications.
12.	ENT_SUBSTITUTE	Replace invalid code unit sequences with a Unicode Replacement Character U+FFFD (UTF-8) or &#xFFFD; (otherwise) instead of returning an empty string.

13.	ENT_DISALLOWED	Replace invalid code points for the given document type with a Unicode Replacement Character U+FFFD (UTF-8) or &#xFFFD; (otherwise) instead of leaving them as is. This may be useful, for instance, to ensure the well-formedness of XML documents with embedded external content.
14.	ENT_HTML401	Handle code as HTML 4.01.
15.	ENT_XML1	Handle code as XML 1.
16.	ENT_XHTML	Handle code as XHTML.
17.	ENT_HTML5	Handle code as HTML 5.
18.	CHAR_MAX	If an array element is equal to CHAR_MAX, no further grouping is done.
19.	LC_CTYPE	For character classification and conversion.
20.	LC_NUMERIC	Used for decimal separator.
21.	LC_TIME	For date and time formatting.
22.	LC_COLLATE	For string comparison.
23.	LC_MONETARY	For localeconv()
24.	LC_ALL	For all LC_ functions.
25.	LC_MESSAGES	For system responses.
26.	STR_PAD_LEFT	Pad left side of an input string.
27.	STR_PAD_RIGHT	Pad right side of an input string.
28.	STR_PAD_BOTH	Pad both sides of an input string.

Math Functions

Math functions are used to handle values within the range of integer and float types.

1.	abs()	Returns the absolute value of a number.
2.	acos()	Returns the arccosine of a number.
3.	acosh()	Returns the inverse hyperbolic cosine of a number.

4.	asin()	Returns the arcsine of a number.
5.	asinh()	Returns the inverse hyperbolic sine of a number.
6.	atan()	Returns the arctangent of a number as a numeric value between: -pi/2 and +pi/2 radians.
7.	atan2()	Returns the angle theta of an (x,y) point as a numeric value between -pi and +pi radians.
8.	atanh()	Returns the inverse hyperbolic tangent of a number.
9.	base_convert()	Converts a number from one base to another.
10.	bindec()	Converts a binary number to a decimal number.
11.	ceil()	Returns the value of a number rounded upwards to the nearest integer.
12.	cos()	Returns the cosine of a number.
13.	cosh()	Returns the hyperbolic cosine of a number.
14.	decbin()	Converts a decimal number to a binary number.
15.	dechex()	Converts a decimal number to a hexadecimal number.
16.	decoct()	Converts a decimal number to an octal number.
17.	deg2rad()	Converts a degree to a radian number.
18.	exp()	Returns the value of E
19.	expm1()	Returns the value of E
20.	floor()	Returns the value of a number rounded downwards to the nearest integer.
21.	fmod()	Returns the remainder(modulo) of the division of the arguments.
22.	getrandmax()	Returns the maximum random number that can be returned by a call to the rand() function.
23.	hexdec()	Converts a hexadecimal number to a decimal number.
24.	hypot()	Returns the length of the hypotenuse of a right-angle triangle.
25.	is_finite()	Returns true if a value is a finite number.
26.	is_infinite()	Returns true if a value is an infinite number.
27.	is_nan()	Returns true if a value is not a number
28.	lcg_value()	Returns a pseudo random number in the range of (0,1)
29.	log()	Returns the natural logarithm(base E) of a number.

30.	log10()	Returns the base-10 logarithm of a number.
31.	log1p()	Returns log(1+number)
32.	max()	Returns the number with the highest value of two specified numbers.
33.	min()	Returns the number with the lowest value of two specified numbers.
34.	mt_getrandmax()	Returns the largest possible value that can be returned by mt_rand()
35.	mt_rand()	Returns a random integer using Mersenne Twister algorithm.
36.	mt_srand()	Seeds the Mersenne Twister random number generator.
37.	octdec()	Converts an octal number to a decimal number.
38.	pi()	Returns the value of pi
39.	pow()	Returns the value is x to the power of y.
40.	rad2deg()	Converts a radian number to a degree
41.	rand()	Returns a random integer
42.	round()	Rounds a number to the nearest integer.
43.	sin()	Returns the sine of a number.
44.	sinh()	Returns the hyperbolic sine of a number.
45.	sqrt()	Returns the sqrt of a number.
46.	srand()	Seeds the random number generator.
47.	tan()	Returns the tangent of an angle.
48.	tanh()	Returns the hyperbolic tangent of an angle.

Math Constants

1.	M_E	Returns e (approx. 2.718)
2.	M_EULER	Returns Euler's constant (approx. 0.577)
3.	M_LNPI	Returns the natural logarithm of PI (approx. 1.144)
4.	M_LN2	Returns the natural logarithm of 2 (approx. 0.693)
5.	M_LN10	Returns the natural logarithm of 10 (approx. 2.302)
6.	M_LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
7.	M_LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
8.	M_PI	Returns PI (approx. 3.14159)

9.	M_PI_2	Returns PI/2 (approx. 1.570)
10.	M_PI_4	Returns PI/4 (approx. 0.785)
11.	M_1_PI	Returns 1/PI (approx. 0.318)
12.	M_2_PI	Returns 2/PI (approx. 0.636)
13.	M_SQRTPI	Returns the square root of PI (approx. 1.772)
14.	M_2_SQRTPI	Returns 2/square root of PI (approx. 1.128)
15.	M_SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
16.	M_SQRT2	Returns the square root of 2 (approx. 1.414)
18	M_SQRT3	Returns the square root of 3 (approx. 1.732)

Date/Time Functions

The date/time functions allow getting the date and time from the server where the PHP script runs. These functions can also be used to format date/time functions as required.

1.	checkdate()	Validates a Gregorian Data
2.	date_default_timezone_get()	Returns the default time zone.
3.	date_default_timezone_set()	Sets the default time zone.
4.	date_sunrise()	Returns the time of sunrise for the time or location.
5.	date_sunset()	Returns the time of sunset for the time or location.
6.	date()	Formats a local time/date.
7.	getdate()	Returns an array that contains the date and time stamp information for a Unix timestamp.
8.	gettimeofday()	Returns an array that contains the current time information.
9.	gmdate()	Formats a GMT/UTC date/time
10.	gmmktime()	Returns the Unix timestamp for a GMT Table.

11.	gmstrftime()	Formats a GMT/UTC time/date according to locale settings.
12.	idate()	Formats a local time/date as an integer.
13.	localtime()	Returns an array that contains the time components of a Unix Timestamp.
14.	microtime()	Returns the current time in microseconds.
15.	mktime()	Returns the Unix timestamp for a date.
16.	strftime()	Formats the local time date according to locale settings.
17.	strptime()	Parses a time/date generated with strftime()
18.	strtotime()	Parses English textual date or time into a Unix timestamp.
19	time()	Returns the current time as a Unix timestamp.

Date/Time Constants

1.	DATE_ATOM	Atom (example: 2005-08-15T16:13:03+0000)
2.	DATE_COOKIE	HTTP Cookies (example: Sun, 14 Aug 2005 16:13:03 UTC)
3.	DATE_ISO8601	ISO-8601 (example: 2005-08-14T16:13:03+0000)
4.	DATE_RFC822	RFC 822 (example: Sun, 14 Aug 2005 16:13:03 UTC)
5.	DATE_RFC850	RFC 850 (example: Sunday, 14-Aug-05 16:13:03 UTC)
6.	DATE_RFC1036	RFC 1036 (example: Sunday, 14-Aug-05 16:13:03 UTC)
7.	DATE_RFC1123	RFC 1123 (example: Sun, 14 Aug 2005 16:13:03 UTC)
8.	DATE_RFC2822	RFC 2822 (Sun, 14 Aug 2005 16:13:03 +0000)
9.	DATE_RSS	RSS (Sun, 14 Aug 2005 16:13:03 UTC)
10.	DATE_W3C	World Wide Web Consortium (example: 2005-08-14T16:13:03+0000)

Calendar Functions

The Calendar extension contains functions that simplify the conversion between two different calendar formats.

1.	cal_days_in_month()	Returns the number of days in a month for a specified year and calendar
2.	cal_from_jd()	Converts a Julian day count into a date of a specified calendar.
3.	cal_info()	Returns information about a given calendar
4.	cal_to_jd()	Converts a date to Julian day count.
5.	easter_date()	Returns the Unix timestamp for midnight on Easter of a specified month.
6.	easter_days()	Returns the number of days after March 21, on which Easter falls for a specified year.
7.	FrenchToJD()	Converts a French Republican date to a Julian day count.
8.	GregorianToJD()	Converts a Gregorian Date to a Julian day count.
9.	JDDayOfWeek()	Returns the day of the week.
10.	JDMonthName()	Returns a month name.
11.	JDToFrench()	Converts a Julian day count to a French Republican date.
12.	JDToGregorian()	Converts a Julian day count to a Gregorian Date.
13.	jdtojewish()	Converts a Julian day count to a Jewish date.
14.	JDToJulian()	Converts a Julian day count to a Julian date.
15.	jdtounix()	Converts a Julian day count to a Unix timestamp.
16.	JewishToJD()	Converts a Jewish date to a Julian day count.
17.	JulianToJD()	Converts a Julian date to a Julian day count.
18.	unixtojd()	Converts a Unix timestamp to a Julian day count.

Calendar Constants

1.	CAL_GREGORIAN	Gregorian Calendar
2.	CAL_JULIAN	Julian Calendar
3.	CAL_JEWISH	Jewish Calendar

4.	CAL_FRENCH	French Republican Calendar.
5.	CAL_NUM_CALS	The number of available calendars.
6.	CAL_DOW_DAYNO	The day of the week as integer, where 0 means Sunday and 6 means Saturday.
7.	CAL_DOW_SHORT	The abbreviated English name of the day of the week.
8.	CAL_DOW_LONG	The English name of the day of the week.
9.	CAL_MONTH_GREGORIAN_SHORT	The abbreviated Gregorian month name.
10.	CAL_MONTH_GREGORIAN_LONG	The Gregorian month name.
11.	CAL_MONTH_JULIAN_SHORT	The abbreviated Julian month name.
12.	CAL_MONTH_JULIAN_LONG	The Julian month name.
13.	CAL_MONTH_JEWISH	The Jewish month name.
14.	CAL_MONTH_FRENCH	The French Republican month name.
15.	CAL_EASTER_DEFAULT	Calculate Easter for years before 1753 according to the Julian calendar, and for later years according to the Gregorian calendar.
16.	CAL_EASTER_ROMAN	Calculate Easter for years before 1583 according to the Julian calendar, and for later years according to the Gregorian calendar.
17.	CAL_EASTER_ALWAYS_GREGORIAN	Calculate Easter according to the proleptic Gregorian calendar.
18.	CAL_EASTER_ALWAYS_JULIAN	Calculate Easter according to the Julian calendar.
19.	CAL_JEWISH_ADD_ALAFIM_GERESH	Adds a geresh symbol (which resembles a single-quote mark) as thousands separator to the year number.
20.	CAL_JEWISH_ADD_ALAFIM	Adds the word alafim as thousands separator to the year number.
21.	CAL_JEWISH_ADD_GERESHAYIM	Add a gershayim symbol (which resembles a double-quote mark) before the final letter of the day and year numbers.

Directory Functions

The directory functions are used to retrieve information about directories and their contents.

1.	chdir()	Changes the current directory
2.	chroot()	Changes the root directory to the current process.
3.	dir()	Opens the directory handle and returns an object.
4.	closedir()	Closes the directory handle.
5.	getcwd()	Returns the current directory.
6.	opendir()	Open a directory handle.
7.	readdir()	Returns an entry from the directory handle.
8.	rewinddir()	Resets a directory handle.
9.	scandir()	Lists the files and directory inside a specified path.

Directory Constants

1.	DIRECTORY_SEPARATOR	In different OS there is different directory separator. In Windows it's \ in Linux it's /. DIRECTORY_SEPARATOR is constant with that OS directory separator.
2.	PATH_SEPARATOR	Semicolon on windows, colon otherwise.

File System Functions

These functions allow to access and manipulate the filesystem.

1.	basename()	Returns the filename component of a path.

2.	chgrp()	Changes the file group.
3.	chmod()	Changes the file mode
4.	chown()	Changes the file owner
5.	clearstatcache()	Clears the file status cache
6.	copy()	Copies a file
7.	delete()	Delete a file
8.	dirname()	Returns the directory name component of a path
9.	disk_free_space() | diskfreespace()	Returns the free space of a directory
10.	disk_total_space()	Returns the total size of a directory
11.	fclose()	Closes an open file
12.	feof()	Tests for end-of-file on an open file
13.	fflush()	Flushes buffered output to an open file.
14.	fgetc()	Returns a character from an open file
15.	fgetcsv()	Checks and parses a line in an open file.
16.	fgets()	Returns a line from an open file.
17.	fgetss()	Returns a line, with HTML and PHP tags removed, from an open file.
18.	file()	Reads a file into an array
19.	file_exists()	Checks whether or not a file or a directory exists.
20.	file_get_contents()	Reads a file into a string
21.	file_put_contents	Writes a string to a file
22.	fileatime()	Returns the last access time of a file
23.	filectime()	Returns the last change time of a file.
24.	filegroup()	Returns the group ID of a file
25.	fileinode()	Returns the inode number of a file
26.	filemtime()	Returns the last modification time of a file
27.	fileowner()	Returns the user ID(owner) of a file
28.	fileperms()	Returns the permissions of a file.

29.	filesize()	Returns the file size.
30.	filetype()	Returns the file type.
31.	flock()	Locks or releases a file.
32.	fnmatch()	Matches a filename or string against a specified pattern
33.	fopen()	Opens a file or URL
34.	fpassthru()	Reads from an open file, until EOF, and writes the result to the output buffer.
35.	fputcsv()	Formats a line as CSV and writes it to an open file.
36.	fputs()	Alias of fwrite.
37.	fread()	Reads from an open file.
38.	fscanf()	Parses input form an open file according to a specified format.
39.	fseek()	Seeks in an open file.
40.	fstat()	Returns information about an open file.
41.	ftell()	Returns the current position in an open file.
42.	ftruncate()	Truncates an open file to a specified length.
43.	fwrite()	Writes to an open file.
44.	glob()	Returns an array of filenames/directories matching a specified pattern.
45.	is_dir()	Checks whether a file is a directory.
46.	is_executable()	Checks whether a file is executable.
47.	is_file()	Checks whether a file is a regular file.
48.	is_link()	Check whether a file is a link.
49.	is_readable()	Checks whether a file is readable.
50.	is_uploaded_file()	Checks whether a file was uploaded via HTTP POST

51.	is_writable()lis_writeable()	Checks whether a file is readable.
52.	link()	Creates a hard link
53.	linkinfo()	Returns information about a hard link.
54.	lstat()	Returns information about a file or symbolic link
55.	mkdir()	Creates a directory.
56.	move_uploaded_file()	Moves an uploaded file to a new location.
57.	parse_ini_file()	Parses a configuration file.
58.	pathinfo()	Returns information about a file path.
59.	pclose()	Closes a pipe opened by popen()
60.	popen()	Opens a pipe
61.	readfile()	Reads a file and writes it to the output buffer.
62.	readlink()	Returns the target of a symbolic link.
63.	realpath()	Returns the absolute pathname.
64.	rename()	Renames a file or directory.
65.	rewind()	Rewinds a file pointer.
66.	rmdir()	Removes an empty directory.
67.	set_file_buffer()	Sets the buffer size if an open file.
68.	stat()	Returns information about a file.
69.	symlink()	Creates a symbolic link
70.	tempnam() | tmpfile()	Creates a unique temporary file
71.	touch()	Sets access and modification time of a file.

72.	umask()	Changes file permissions for files.
73.	unlink()	Deletes a file.

File System Constants

1.	GLOB_BRACE	Expands to match 'a', 'b', or 'c'
2.	GLOB_ONLYDIR	Return only directory entries which match the pattern
3.	GLOB_MARK	Adds a slash to each directory returned
4.	GLOB_NOSORT	Return files as they appear in the directory (no sorting). When this flag is not used, the path names are sorted alphabetically
5.	GLOB_NOCHECK	Return the search pattern if no files matching it were found
6.	GLOB_NOESCAPE	Backslashes do not quote metacharacters.
7.	GLOB_ERR	Stop on read errors (like unreadable directories), by default errors are ignored.
8.	PATHINFO_DIRNAME	Returns information about the path directory.
9.	PATHINFO_BASENAME	Returns information about the path basename.
10.	PATHINFO_EXTENSION	Returns information about the path extension.
11.	FILE_USE_INCLUDE_PATH	Search for the file in the include path.
12.	FILE_APPEND	Append a file to a file. The target file needs to be writable for this.
13.	FILE_IGNORE_NEW_LINES	Omit newline at the end of each array element.
14.	FILE_SKIP_EMPTY_LINES	Skips empty lines.

Zip Functions

These functions allow you to read zip files. The ZIP extension requires libzip which is a C library for reading, creating and modifying zip archives.

1.	zip_close()	Closes a ZIP File
2.	zip_entry_close()	Closes an entry in the ZIP file

3.	zip_entry_compressedsize()	Returns compressed size of an entry in the ZIP file.
4.	zip_entry_compressionmethod()	Returns the compression method of an entry in the ZIP File.
5.	zip_entry_filesize()	Returns the actual file size of an entry in the ZIP File.
6.	zip_entry_name()	Returns the name of an entry in the ZIP File.
7.	zip_entry_open()	Opens an entry in the ZIP File for reading.
8.	zip_entry_read()	Reads from an open entry in the ZIP File.
9.	zip_open()	Opens a ZIP File
10.	zip_read()	Reads the next entry in the ZIP File.

Filter Functions

These functions are used to validate and filter data coming from insecure input sources. These filter functions are enabled by default.

1.	filter_has_var()	Checks if a variable of a specified input type exist.
2.	filter_id()	Returns the ID number of a specified filter.
3.	filter_input()	Get input from outside the script and filter it.
4.	filter_input_array()	Get multiple inputs from outside the script and filters them.
5.	filter_list()	Returns an array of all supported filters
6.	filter_var_array()	Get multiple variables and filter them.
7.	filter_var()	Get a variable and filter it.

Filter Constants

1.	FILTER_DEFAULT	Do nothing, optionally strip/encode special characters. Equivalent to FILTER_UNSAFE_RAW
2.	FILTER_FLAG_NONE	Allows no flags

3.	FILTER_FLAG_ALLOW_OCTAL	Only for inputs that starts with a zero (0) as octal numbers. This only allows the succeeding digits to be 0-7
4.	FILTER_FLAG_ALLOW_HEX	Only for inputs that starts with 0x/0X as hexadecimal numbers. This only allows succeeding characters to be a-fA-F0-9
5.	FILTER_FLAG_STRIP_LOW	Strip characters with ASCII value lower than 32
6.	FILTER_FLAG_STRIP_HIGH	Strip characters with ASCII value greater than 127
7.	FILTER_FLAG_ENCODE_LOW	Encode characters with ASCII value lower than 32
8.	FILTER_FLAG_ENCODE_HIGH	Encode characters with ASCII value greater than 127
9.	FILTER_FLAG_ENCODE_AMP	Encode &
10.	FILTER_FLAG_NO_ENCODE_QUOTES	Do not encode ' and "
11.	FILTER_FLAG_EMPTY_STRING_NULL	Not in use
12.	FILTER_FLAG_ALLOW_FRACTION	Allows a period (.) as a fractional separator in numbers
13.	FILTER_FLAG_ALLOW_THOUSAND	Allows a comma (,) as a thousands separator in numbers
14.	FILTER_FLAG_ALLOW_SCIENTIFIC	Allows an e or E for scientific notation in numbers
15.	FILTER_FLAG_PATH_REQUIRED	The URL must contain a path part
16.	FILTER_FLAG_QUERY_REQUIRED	The URL must contain a query string
17.	FILTER_FLAG_IPV4	Allows the IP address to be in IPv4 format
18.	FILTER_FLAG_IPV6	Allows the IP address to be in IPv6 format

19.	FILTER_FLAG_NO_RES_RANGE	Fails validation for the reserved IPv4 ranges: 0.0.0.0/8, 169.254.0.0/16, 127.0.0.0/8 and 240.0.0.0/4, and for the reserved IPv6 ranges: ::1/128, ::/128, ::ffff:0:0/96 and fe80::/10
20.	FILTER_FLAG_NO_PRIV_RANGE	Fails validation for the private IPv4 ranges: 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16, and for the IPv6 addresses starting with FD or FC
21.	FILTER_FLAG_EMAIL_UNICODE	Allows the local part of the email address to contain Unicode characters
22.	FILTER_REQUIRE_SCALAR	The value must be a scalar
23.	FILTER_REQUIRE_ARRAY	The value must be an array
24.	FILTER_FORCE_ARRAY	Treats a scalar value as an array with the scalar value as only element
25.	FILTER_NULL_ON_FAILURE	Return NULL on failure for unrecognized boolean values
26.	FILTER_VALIDATE_BOOLEAN	Validates a boolean
27.	FILTER_VALIDATE_EMAIL	Validates value as a valid email address
28.	FILTER_VALIDATE_FLOAT	Validates value as float
29.	FILTER_VALIDATE_INT	Validates value as integer
30.	FILTER_VALIDATE_IP	Validates value as IP address
31.	FILTER_VALIDATE_MAC	Validates value as MAC address
32.	FILTER_VALIDATE_REGEXP	Validates value against a regular expression
33.	FILTER_VALIDATE_URL	Validates value as URL

34.	FILTER_SANITIZE_EMAIL	Removes all illegal characters from an email address
35.	FILTER_SANITIZE_ENCODED	Removes/Encodes special characters
36.	FILTER_SANITIZE_MAGIC_QUOTES	Apply addslashes()
37.	FILTER_SANITIZE_NUMBER_FLOAT	Remove all characters, except digits, +- signs, and optionally .,eE
38.	FILTER_SANITIZE_NUMBER_INT	Removes all characters except digits and + - signs
39.	FILTER_SANITIZE_SPECIAL_CHARS	Removes special characters
40.	FILTER_SANITIZE_STRING	Removes tags/special characters from a string
41.	FILTER_SANITIZE_STRIPPED	Alias of FILTER_SANITIZE_STRING
42.	FILTER_SANITIZE_URL	Removes all illegal characters from s URL
43.	FILTER_UNSAFE_RAW	Do nothing, optionally strip/encode special characters
44.	FILTER_CALLBACK	Call a user-defined function to filter data
45.	INPUT_POST	POST variables
46.	INPUT_GET	GET variables
47.	INPUT_COOKIE	COOKIE variables
48.	INPUT_ENV	ENV variables
49.	INPUT_SERVER	SERVER variables

Mail Functions

The Mail function allows sending mail directly from script.

1.	ezmlm_hash()	Calculates the hash value by the EZMLM mailing list system.
2.	mail()	Allows you to send emails directly from a script.

FTP Functions

The FTP Functions give the client access to the file servers through the File Transfer Protocol(FTP). The FTP functions are used to open, login and close connections, upload, download, rename, delete and get information on files from file servers.

1.	ftp_alloc()	Allocates space for a file to be uploaded to the FTP server.
2.	ftp_cdup()	Changes the current directory to the parent directory on the FTP server.
3.	ftp_chdir()	Changes the current directory on the FTP server
4.	ftp_chmod()	Sets the permissions on a file via FTP
5.	ftp_close()	Closes an FTP connection.
6.	ftp_connect()	Opens an FTP connection
7.	ftp_delete()	Deletes a file on the FTP server.
8.	ftp_exec()	Executes a program/command on the FTP server.
9.	ftp_fget()	Downloads a file from the FTP server and saves it to an open file.
10.	ftp_fput()	Uploads from an open file and saves it to a file on the FTP server.
11.	ftp_get_option()	Returns runtime behaviours of the FTP connection.
12.	ftp_get()	Downloads a file from the FTP server.
13.	ftp_login()	Logs on to an FTP connection
14.	ftp_mdtm()	Returns the last modified time of a specified file.
15.	ftp_mkdir()	Creates a new directory on the FTP server.
16.	ftp_nb_continue()	Continues retrieving/sending a file (non-blocking)
17.	ftp_nb_fget()	Downloads a file from the FTP server and saves it to an open file (non-blocking)
18.	ftp_nb_fput()	Uploads from an open file and saves it to a file on the FTP server (non-blocking)
19.	ftp_nb_get()	Downloads a file from the FTP server (non-blocking)
20.	ftp_nb_put()	Uploads a file to the FTP server (non-blocking)

21.	ftp_nlist()	Lists the files in a specified directory on the FTP server.
22.	ftp_pasv()	Turns passive mode on or off.
23.	ftp_put()	Uploads a file to the FTP server.
24.	ftp_pwd()	Returns the current directory name.
25.	ftp_quit()	Alias of ftp_close()
26.	ftp_raw()	Sends a raw command to the FTP server
27.	ftp_rawlist()	Returns a detailed list of files in the specified directory
28.	ftp_rename()	Renames a file or a directory on the FTP server.
29.	ftp_rmdir()	Removes a directory on the FTP server.
30.	ftp_set_option()	Sets runtime options for the FTP connection
31.	ftp_site()	Sends a SITE command to the server.
32.	ftp_size()	Returns the size of the specified file.
33.	ftp_ssl_connect()	Opens a secure SSL-FTP connection
34.	ftp_systype()	Returns the system type identifier of the FTP server.

FTP Constants

1.	FTP_ASCII | FTP_TEXT	It is the FTP transfer mode.
2.	FTP_BINARY | FTP_IMAGE	It is the FTP transfer mode.
3.	FTP_TIMEOUT_SEC	Timeout used for network operations
4.	FTP_AUTOSEEK	Sets miscellaneous runtime FTP options.
5.	FTP_AUTORESUME	Automatically determine the resume position and start position for GET and PUT requests. FTP_AUTOSEEK needs to be enabled.
6.	FTP_FAILED	Asynchronous transfer has failed.
7.	FTP_FINISHED	Asynchronous transfer has completed.
8.	FTP_MOREDATA	Asynchronous transfer is in progress.

HTTP Functions

1.	header()	Sends a raw HTTP header to a client.
2.	headers_list()	Returns a list of response headers sent(or ready to send)

3.	headers_sent()	Checks if/where the HTTP headers have been sent.
4.	setcookie()	Sends an HTTP cookie to a client.
5.	setrawcookie()	Sends an HTTP cookie without URL encoding the cookie value.

Simple XML Functions

Allow us to easily manipulate and get XML Data. It gets an element's name, attributes and textual content once the XML document's structure or layout is known. Simple XML converts an XML document into a data structure that can be iterated through like arrays and objects.

1.	__construct()	Creates a new SimpleXML Element Object
2.	addAttribute()	Adds an attribute to the SimpleXML element
3.	addChild()	Adds a child element from the SimpleXML element.
4.	asXML()	Gets an XML string from a SimpleXML element.
5.	attributes()	Gets a SimpleXML element's attributes.
6.	children()	Gets the children of a specified node.
7.	getDocNamespaces()	Gets the namespaces of an XML document.
8.	getName()	Gets the name of a SimpleXML element.
9.	getNamespace()	Gets the namespaces from XML data.
10.	registerXPathNamespace()	Creates a namespaces context for the next XPath query.
11.	simplexml_import_dom()	Gets a SimpleXMLElement object from a DOM node.
12.	simplexml_load_file()	Gets a SimpleXMLElement object from an XML document.
13.	simplexml_load_string()	Gets a SimpleXMLElement object from an XML string.
14.	xpath()	Runs an XPath query on XML data.

LIBXML Functions

These functions are used with SimpleXML, XSLT and DOM Functions.

1.	libxml_clear_errors()	Clear libxml error buffer
2.	libxml_get_errors()	Retrieve array of errors
3.	libxml_get_last_error()	Retrieve last error from libxml
4.	libxml_set_streams_context()	Set the streams context for the next libxml document load or write.
5.	libxml_use_internal_errors()	Disable libxml errors and allow user to fetch error information as needed.

LIBXML Constants

1.	LIBXML_BIGLINES	Make line numbers greater than 65535 to be reported correctly
2.	LIBXML_COMPACT	Set small nodes allocation optimization. This may improve the application performance
3.	LIBXML_DOTTED_VERSION	Get dotted libxml version (e.g. 2.6.5 or 2.6.17)
4.	LIBXML_DTDATTR	Set default DTD attributes
5.	LIBXML_DTDLOAD	Load external subset
6.	LIBXML_DTDVALID	Validate with the DTD
7.	LIBXML_ERR_ERROR	Get recoverable errors
8.	LIBXML_ERR_FATAL	Get fatal errors
9.	LIBXML_ERR_NONE	Get no errors
10.	LIBXML_ERR_WARNING	Get simple warnings
11.	LIBXML_HTML_NOIMPLIED	Set HTML_PARSE_NOIMPLIED flag. This turns off automatic adding of implied html/body elements
12.	LIBXML_HTML_NODEFDTD	Set HTML_PARSE_NODEFDTD flag. This prevents a default doctype to be added, if no doctype is found
13.	LIBXML_NOBLANKS	Remove blank nodes
14.	LIBXML_NOCDATA	Set CDATA as text nodes

15.	LIBXML_NOEMPTYTAG	Change empty tags (e.g. to </br>), only available in the DOMDocument->save() and DOMDocument->saveXML() functions
16.	LIBXML_NOENT	Substitute entities
17.	LIBXML_NOERROR	Do not show error reports
18.	LIBXML_NONET	Stop network access while loading documents
19.	LIBXML_NOWARNING	Do not show warning reports
20.	LIBXML_NOXMLDECL	Drop the XML declaration when saving a document
21.	LIBXML_NSCLEAN	Remove excess namespace declarations
22.	LIBXML_PARSEHUGE	Set XML_PARSE_HUGE flag. This relaxes any hardcoded limit from the parser, such as maximum depth of a document or the size of text nodes
23.	LIBXML_PEDANTIC	Set XML_PARSE_PEDANTIC flag. This enables pedantic error reporting
24.	LIBXML_XINCLUDE	Use XInclude substitution
25.	LIBXML_VERSION	Get libxml version (e.g. 20605 or 20617)
26.	LIBXML_SCHEMA_CREATE	Create default or fixed value nodes during XSD schema validation

XML Parser Functions

XML is a data format intended for standardized structured document exchange. They help to parse XML documents. They however do not help to validate the documents. The XML parser functions enables us to create XML parsers and define handlers for XML events.

1.	utf8_decode()	Decodes an UTF-8 string to ISO-8859-1
2.	utf8_encode()	Encodes an ISO-8859-1 string to UTF-8

3.	xml_error_string()	Gets an error string from the XML parser.
4.	xml_get_current_byte_index()	Gets the current byte index from the XML parser.
5.	xml_get_current_column_number()	Gets the current column number from the XML parser.
6.	xml_get_current_line_number()	Gets the current line number from the XML parser
7.	xml_get_error_code()	Gets an error code from the XML parser.
8.	xml_parse()	Parses an XML document.
9.	xml_parse_into_struct()	Parse XML data into an array.
10.	xml_parser_create_ns()	Create an XML parser with namespace support.
11.	xml_parser_create()	Create an XML parser
12.	xml_parser_free()	Free an XML parser
13.	xml_parser_get_option()	Get options from an XML parser
14.	xml_parser_set_option()	Set options in an XML parser
15.	xml_set_character_data_handler()	Set handler function for character data.
16.	xml_set_default_handler()	Set default handler function.
17.	xml_set_element_handler()	Set handler function for start and end of element of elements.
18.	xml_set_end_namespace_decl_handler()	Set handler function for the end of namespace declarations
19.	xml_set_external_entity_ref_handler()	Set handler functions for external entities.
20.	xml_set_notation_decl_handler()	Set handler function for notation declarations.
21.	xml_set_object()	Use XML Parser within an object.
22.	xml_set_processing_instruction_handler()	Set handler function for processing instruction.
23.	xml_set_start_namespace_decl_handler()	Set handler function for the start of namespace declarations
24.	xml_set_unparsed_entity_decl_handler()	Set handler function for unparsed entity declarations.

XML Parser Constants

1.	XML_ERROR_NONE (integer)	2.	XML_ERROR_ASYNC_ENTITY (integer)
3.	XML_ERROR_NO_MEMORY (integer)	4.	XML_ERROR_BAD_CHAR_REF (integer)
5.	XML_ERROR_SYNTAX (integer)	6.	XML_ERROR_BINARY_ENTITY_REF (integer)
7.	XML_ERROR_NO_ELEMENTS (integer)	8.	XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF (integer)
9.	XML_ERROR_INVALID_TOKEN (integer)	10.	XML_ERROR_MISPLACED_XML_PI (integer)

11.	XML_ERROR_UNCLOSED_TOKEN (integer)	12.	XML_ERROR_UNKNOWN_ENCODING (integer)
13.	XML_ERROR_PARTIAL_CHAR (integer)	14.	XML_ERROR_INCORRECT_ENCODING (integer)
15.	XML_ERROR_TAG_MISMATCH (integer)	16.	XML_ERROR_UNCLOSED_CDATA_SECTION (integer)
17.	XML_ERROR_DUPLICATE_ATTRIBUTE (integer)	18.	XML_ERROR_EXTERNAL_ENTITY_HANDLING (integer)
19.	XML_ERROR_JUNK_AFTER_DOC_ELEMENT (integer)	20.	XML_OPTION_CASE_FOLDING (integer)
21.	XML_ERROR_PARAM_ENTITY_REF (integer)	22.	XML_OPTION_TARGET_ENCODING (integer)
23.	XML_ERROR_UNDEFINED_ENTITY (integer)	24.	XML_OPTION_SKIP_TAGSTART (integer)
25.	XML_ERROR_RECURSIVE_ENTITY_REF (integer)	26.	XML_OPTION_SKIP_WHITE (integer)

MYSQLI Functions

These functions allow you to access MySQL database servers.

1.	mysqli::$affected_rows	Gets the number of affected rows in the previous MySQL operation
2.	mysqli::autocommit()	Turns on or off auto committing database modifications
3.	mysqli::change_user()	Changes the user of the specified database connection.
4.	mysqli::character_set_name()	Returns the default character set for the database connection
5.	mysqli::$client_info	Get MySQL client info
6.	mysqli::$client_version	Returns the MySQL client version as a string.
7.	mysqli::close()	Closes a previously opened database connection.
8.	mysqli::commit()	Commits the current transaction.
9.	mysqli::$connect_errno	Returns the error code from the last connect call.
10.	mysqli::$connect_error	Returns a string description of the last connect error.

11.	mysqli::__construct()	Open a new connection to the MySQL server.
12.	mysqli::debug()	Performs debugging operations.
13.	mysqli::dump_debug_info()	Dump debugging information into the log.
14.	mysqli::$errno	Returns the error code for the most recent function call.
15.	mysqli::$error_list	Returns a list of errors from the last command executed.
16.	mysqli::$error	Returns a string description of the last error.
17.	mysqli::$field_count	Returns the number of columns for the most recent query.
18.	mysqli::get_charset()	Returns a character set object.
19.	mysqli::get_client_info()	Get MySQL client info.
20.	mysqli_get_client_stats()	Returns client per-process statistics
21.	mysqli_get_client_version()	Returns the MySQL client version as a string.
22.	mysqli::get_connection_stats()	Returns statistics about the client connection
23.	mysqli::$host_info	Returns a string representing the type of connection used.
24.	mysqli::$protocol_version	Returns the version of the MySQL protocol used.
25.	mysqli::$server_info	Returns the version of the MySQL Server.
26.	mysqli::$server_version	Returns the version of the MySQL server as an integer.
27.	mysqli::get_warnings()	Get the result of “SHOW WARNINGS”.
28.	mysqli::$info	Retrieves information about the most recently executed query.
29.	mysqli::init()	Initializes MYSQLi and returns a resource for use with mysqli_real connect()
30.	mysqli::$insert_id	Returns the auto generated id used in the last query.
31.	mysqli::kill()	Asks the server to kill a MySQL thread.
32.	mysqli::more_results()	Check if there are any more query results from a multi query.
33.	mysqli::multi_query()	Performs a query on a database.

34.	mysqli::next_result()	Prepare next result from a multi_query.
35.	mysqli::options()	Set options.
36.	mysqli::ping()	Pings a server connection, or tries to reconnect if the condition has gone down.
37.	mysqli::poll()	Poll connections.
38.	mysqli::prepare()	Prepare a SQL statement for execution.
39.	mysqli::query()	Performs a query on the database.
40.	mysqli::real_connect()	Opens a connection to a mysql server.
41.	mysqli::real_escape_string()	Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.
42.	mysqli::real_query()	Execute a SQL query.
43.	mysqli::reap_async_query()	Get the result from async query.
44.	mysqli::refresh()	Refreshes
45.	mysqli::rollback()	Rolls back current transaction.
46.	mysqli::rpl_query_type()	Returns RPL query type.
47.	mysqli::select_db()	Selects the default database for database queries.
48.	mysqli::send_query()	Send the query and return.
49.	mysqli::set_charset()	Sets the default client character set.
50.	mysqli::set_local_infile_default()	Unsets user defined handler for load local infile command.

51.	mysqli::set_local_infile_handler()	Se callback function for LOAD DATA LOCAL INFILE command.
52.	mysqli::$sqlstate	Returns the SQLSTATE error from previous MySQL operation.
53.	mysqli::ssl_set()	Used for establishing secure connections using SSL
54.	mysqli::stat()	Gets the current system status.
55.	mysqli::stmt_init()	Initializes a statement and returns an object for use with mysqli_stmt prepare.
56.	mysqli::store_result()	Transfers a result set from the last query.
57.	mysqli::$thread_id	Returns the thread ID for the current connection.
58.	mysqli::thread_safe()	Returns whether thread safety is given or not.
59.	mysqli::use_result()	Initiate a result set retrieval.
60.	mysqli::$warning_count	Returns the number of warnings from the last query for the given link.

MYSQLI Statements Constants

1.	mysqli_stmt::$affected_rows	Returns the total number of rows changed, deleted, or inserted by the last executed statement.
2.	mysqli_stmt::attr_get()	Used to get the current value of a statement attribute.
3.	mysqli_stmt::attr_set()	Used to modify the behaviour of a prepared statement.
4.	mysqli_stmt::bind_param()	Binds variables to a prepared statement as parameters.
5.	mysqli_stmt::bind_result()	Binds variables to a prepared statement for result storage.
6.	mysqli_stmt::close()	Closes a prepared statement.
7.	mysqli_stmt::data_seek()	Seeks to an arbitrary row in statement result set.
8.	mysqli_stmt::$errno	Returns the error code for the most recent statement call.
9.	mysqli_stmt::$error_list	Returns a list of errors from the last statement executed.
10.	mysqli_stmt::$error	Returns a string description for last statement error.
11.	mysqli_stmt::execute()	Executes a prepared query.
12.	mysqli_stmt::fetch()	Fetch results from a prepared statement into the bound variables.
13.	mysqli_stmt::$field_count	Returns the number of fields in the given statement.
14.	mysqli_stmt::free_result()	Frees stored result memory for the given statement handle.

15.	mysqli_stmt::get_result()	Gets a result from a prepared statement.
16.	mysqli_stmt::get_warnings()	Get result of SHOW WARNINGS
17.	mysqli_stmt::$insert_id	Get the ID generated from the previous insert operation.
18.	mysqli_stmt::more_results()	Check if there are any more query results from a multiple query.
19.	mysqli_stmt::next_result()	Reads the next result from a multiple query.
20.	mysqli_stmt::$num_rows	Return the number of rows in statements result set.
21.	mysqli_stmt::$param_count	Returns the number of parameters for the given statement.
22.	mysqli_stmt::prepare()	Prepares an SQL statement for execution.
23.	mysqli_stmt::reset()	Resets a prepared statement.
24.	mysqli_stmt::result_metadata()	Returns result set metadata from a prepared statement.
25.	mysqli_stmt::send_long_data()	Send data in blocks.
26.	mysqli_stmt::$sqlstate	Returns SQLSTATE error from previous statement operation.
27.	mysqli_stmt::store_result()	Transfers a result set from a prepared statement.

MYSQLI Result Class

1.	mysqli_result::$current_field	Get current field offset of a result pointer.
2.	mysqli_result::data_seek()	Adjusts the result pointer to an arbitrary row in the result.
3.	mysqli_result::fetch_all()	Fetches all result rows as an associative array, a numeric array, or both.
4.	mysqli_result::fetch_array()	Fetch a result row as an associative, a numeric array, or both.
5.	mysqli_result::fetch_assoc()	Fetch a result row as an associative array.
6.	mysqli_result::fetch_field_direct()	Fetch meta-data for a single field.
7.	mysqli_result::fetch_field()	Returns the next field in the result set.
8.	mysqli_result::fetch_fields()	Returns an array of objects representing the fields in a result set.
9.	mysqli_result::fetch_object()	Returns the current row of a result se as an object.
10.	mysqli_result::fetch_row()	Get a result row as an enumerated array.
11.	mysqli_result::$field_count	Get the number of fields in a result.

12.	mysqli_result::field_seek()	Set result pointer to a specified field offset.
13.	mysqli_result::free()	Frees the memory associated with a result.
14.	mysqli_result::$lengths	Returns the length of the column of the current row in the result set.
15.	mysqli_result::$num_rows	Gets the number of rows in a result.

MYSQLI Driver Class

1.	mysqli_driver::embedded_server_end()	Stop embedded server
2.	mysqli_driver::embedded_server_start()	Initialize and start embedded server
3.	mysqli_driver::$report_mode	Enables or disables internal report functions
4.	mysqli_driver::$client_info	The Clients API header version.
5.	mysqli_driver::$client_version	The Client Version
6.	mysqli_driver::$driver_version	The MySQLi Driver Version
7.	mysqli_driver::$embedded	Whether MySQLi Embedded support is enabled.
8.	mysqli_driver::$reconnect	Allow or prevent reconnect
9.	mysqli_driver::$report_mode	Set to MYSQLi_REPORT_OFF, MYSQLi_REPORT_ALL or any combination of MYSQLI_REPORT_STRICT (throw Exceptions for errors), MYSQLI_REPORT_ERROR (report errors) and MYSQLI_REPORT_INDEX (errors regarding indexes).

MYSQLI Warning Class

1.	mysqli_warning::__construct()	The _construct purpose
2.	mysqli_warning::next()	The next purpose
3.	mysqli_warning::$message	Message string
4.	mysqli_warning::$sqlstate	SQL State
5.	mysqli_warning::$errno	The MySQLi Driver Version

MYSQLi Constants

1.	MYSQLI_READ_DEFAULT_GROUP	Read options from the named group from my.cnf or the file specified with MYSQLI_READ_DEFAULT_FILE

2.	MYSQLI_READ_DEFAULT_FILE	Read options from the named option file instead of from my.cnf
3.	MYSQLI_OPT_CONNECT_TIMEOUT	Connect timeout in seconds
4.	MYSQLI_OPT_LOCAL_INFILE	Enables command LOAD LOCAL INFILE
5.	MYSQLI_INIT_COMMAND	Command to execute when connecting to MySQL server. Will automatically be re-executed when reconnecting.
6.	MYSQLI_CLIENT_SSL	Use SSL (encrypted protocol). This option should not be set by application programs; it is set internally in the MySQL client library
7.	MYSQLI_CLIENT_COMPRESS	Use compression protocol
8.	MYSQLI_CLIENT_INTERACTIVE	Allow interactive_timeout seconds (instead of wait_timeout seconds) of inactivity before closing the connection. The client's session wait_timeout variable will be set to the value of the session interactive_timeout variable.
9.	MYSQLI_CLIENT_IGNORE_SPACE	Allow spaces after function names. Makes all functions names reserved words.
10.	MYSQLI_CLIENT_NO_SCHEMA	Don't allow the db_name.tbl_name.col_name syntax.
11.	MYSQLI_CLIENT_MULTI_QUERIES	Allows multiple semicolon-delimited queries in a single mysqli_query() call.
12.	MYSQLI_STORE_RESULT	For using buffered resultsets
13.	MYSQLI_USE_RESULT	For using unbuffered result sets
14.	MYSQLI_ASSOC	Columns are returned into the array having the fieldname as the array index.
15.	MYSQLI_NUM	Columns are returned into the array having an enumerated index.

16.	MYSQLI_BOTH	Columns are returned into the array having both a numerical index and the fieldname as the associative index.
17.	MYSQLI_NOT_NULL_FLAG	Indicates that a field is defined as NOT NULL
18.	MYSQLI_PRI_KEY_FLAG	Field is part of a primary index
19.	MYSQLI_UNIQUE_KEY_FLAG	Field is part of a unique index.
20.	MYSQLI_MULTIPLE_KEY_FLAG	Field is part of an index.
21.	MYSQLI_BLOB_FLAG	Field is defined as BLOB
22.	MYSQLI_UNSIGNED_FLAG	Field is defined as UNSIGNED
23.	MYSQLI_ZEROFILL_FLAG	Field is defined as ZEROFILL
24.	MYSQLI_AUTO_INCREMENT_FLAG	Field is defined as AUTO_INCREMENT
25.	MYSQLI_TIMESTAMP_FLAG	Field is defined as TIMESTAMP
26.	MYSQLI_SET_FLAG	Field is defined as SET
27.	MYSQLI_NUM_FLAG	Field is defined as NUMERIC
28.	MYSQLI_PART_KEY_FLAG	Field is part of an multi-index
29.	MYSQLI_GROUP_FLAG	Field is part of GROUP BY
30.	MYSQLI_TYPE_DECIMAL	Field is defined as DECIMAL
31.	MYSQLI_TYPE_NEWDECIMAL	Precision math DECIMAL or NUMERIC field (MySQL 5.0.3 and up)
32.	MYSQLI_TYPE_BIT	Field is defined as BIT (MySQL 5.0.3 and up)
32.	MYSQLI_TYPE_TINY	Field is defined as TINYINT
34.	MYSQLI_TYPE_SHORT	Field is defined as SMALLINT
35.	MYSQLI_TYPE_LONG	Field is defined as INT
36.	MYSQLI_TYPE_FLOAT	Field is defined as FLOAT
37.	MYSQLI_TYPE_DOUBLE	Field is defined as DOUBLE
38.	MYSQLI_TYPE_NULL	

39.	MYSQLI_TYPE_TIMESTAMP	Field is defined as TIMESTAMP
40.	MYSQLI_TYPE_LONGLONG	Field is defined as BIGINT
41.	MYSQLI_TYPE_INT24	Field is defined as MEDIUMINT
42.	MYSQLI_TYPE_DATE	Field is defined as DATE
43.	MYSQLI_TYPE_TIME	Field is defined as TIME
44.	MYSQLI_TYPE_DATETIME	Field is defined as DATETIME
45.	MYSQLI_TYPE_YEAR	Field is defined as YEAR
46.	MYSQLI_TYPE_NEWDATE	Field is defined as DATE
47.	MYSQLI_TYPE_INTERVAL	Field is defined as INTERVAL
48.	MYSQLI_TYPE_ENUM	Field is defined as ENUM
49.	MYSQLI_TYPE_SET	Field is defined as SET
50.	MYSQLI_TYPE_TINY_BLOB	Field is defined as TINYBLOB
51.	MYSQLI_TYPE_MEDIUM_BLOB	Field is defined as MEDIUMBLOB
52.	MYSQLI_TYPE_LONG_BLOB	Field is defined as LONGBLOB
53.	MYSQLI_TYPE_BLOB	Field is defined as BLOB
54.	MYSQLI_TYPE_VAR_STRING	Field is defined as VARCHAR
55.	MYSQLI_TYPE_STRING	Field is defined as CHAR or BINARY
56.	MYSQLI_TYPE_CHAR	Field is defined as TINYINT. For CHAR, see MYSQLI_TYPE_STRING

57.	MYSQLI_TYPE_GEOMETRY	Field is defined as GEOMETRY
58.	MYSQLI_NEED_DATA	More data available for bind variable
59.	MYSQLI_NO_DATA	No more data available for bind variable
60.	MYSQLI_DATA_TRUNCATED	Data truncation occurred. Available since PHP 5.1.0 and MySQL 5.0.5.
61.	MYSQLI_ENUM_FLAG	Field is defined as ENUM. Available since PHP 5.3.0.
62.	MYSQLI_REPORT_INDEX	Report if no index or bad index was used in a query.
63.	MYSQLI_REPORT_ERROR	Report errors from mysqli function calls.
64.	MYSQLI_REPORT_STRICT	Throw a mysqli_sql_exception for errors instead of warnings.
65.	MYSQLI_REPORT_ALL	Set all options on (report all).
66.	MYSQLI_REPORT_OFF	Turns reporting off.
67.	MYSQLI_DEBUG_TRACE_ENABLED	Is set to 1 if mysqli_debug() functionality is enabled.
68.	MYSQLI_SERVER_QUERY_NO_GOOD_INDEX_USED	
69.	MYSQLI_SERVER_QUERY_NO_INDEX_USED	
70.	MYSQLI_REFRESH_GRANT	Refreshes the grant tables.
71.	MYSQLI_REFRESH_LOG	Flushes the logs, like executing the FLUSH LOGS SQL statement.
72.	MYSQLI_REFRESH_TABLES	Flushes the table cache, like executing the FLUSH TABLES SQL statement.
73.	MYSQLI_REFRESH_HOSTS	Flushes the host cache, like executing the FLUSH HOSTS SQL statement.
74.	MYSQLI_REFRESH_STATUS	Reset the status variables, like executing the FLUSH STATUS SQL statement.
75.	MYSQLI_REFRESH_THREADS	Flushes the thread cache.

76.	MYSQLI_REFRESH_SLAVE	On a slave replication server: resets the master server information, and restarts the slave. Like executing the RESET SLAVE SQL statement.
77.	MYSQLI_REFRESH_MASTER	On a master replication server: removes the binary log files listed in the binary log index, and truncates the index file. Like executing the RESET MASTER SQL statement.

MISC Functions

All other categorized functions have been placed under this category. The behavior of these functions are affected by settings in the php.ini file.

1.	connection_aborted()	Checks whether the client has disconnected.
2.	connection_status()	Returns the current connection status
3.	connection_timeout()	Deprecated since PHP 4.0.5
4.	constant()	Returns the value of a constant.
5.	define()	Defines a constant.
6.	defined()	Checks whether a constant exists.
7.	die()	Prints a message and exits the current script
8.	eval()	Evaluates a string as PHP Code
9.	exit()	Prints a message and exits the current script.
10.	get_browser()	Returns the capabilities of the user's browser
11.	highlight_file()	Outputs a file with the PHP syntax highlighted
12.	highlight_string()	Outputs a string with the PHP syntax highlighted.
13.	ignore_user_abort()	Sets whether a remote client can abort the running of a script.
14.	pack()	Packs data into a binary string

15.	php_strip_whitespace()	Removes the whitespaces and PHP Comments from the source code of a file.
16.	show_source()	Alias of highlight_file()
17.	sleep()	Delays code execution for a number of seconds.
18.	time_nanosleep()	Delays code execution for a number of seconds and nanoseconds.
19.	time_sleep_until()	Delays code execution until a specified time.
20.	uniqid()	Generates a unique ID
21.	unpack()	Unpacks data from a binary string.
22.	usleep()	Delays code execution for a number of microseconds.

MISC. Constants

1.	CONNECTION_ABORTED	Connection is aborted by the user or network error.
2.	CONNECTION_NORMAL	Connection is running normally.
3.	CONNECTION_TIMEOUT	Connection timed out.
4.	__COMPILER_HALT_OFFSET__	Determines the byte position of the data start.

Object-Oriented Programming

PHP codes can also be written in Object-Oriented Format. OOP is faster and easier to execute. The OOP structure makes the code easier to maintain, modify and debug.

1.	__construct()	Classes with a constructor method call this method on each newly created object, so it is suitable for any initialization that the object may need before it is used.
2.	__destruct()	Called as soon as there are no other references to a particular object, or in any order during the shutdown sequence.
3.	__callStatic()	Triggered when invoking inaccessible methods in a static context.

4.	__call()	Triggered when invoking inaccessible methods in an object context.
5.	__get()	Utilized for reading data from inaccessible properties.
6.	__set()	Run when writing data to inaccessible properties.
7.	__isset()	Triggered by calling isset() or empty() on inaccessible properties.
8.	__unset()	Invoked when unset() is used on inaccessible properties.
9.	__sleep()	The intended use of _sleep() is to commit pending data or perform similar cleanup tasks. Also, the function is useful if you have very large objects which do not need to be saved completely.
10.	__wakeup()	Used to re-establish database connections that may have been lost during serialization and perform other reinitialization tasks.
11.	__toString()	Allows a class to decide how it will react when it is treated like a string.
12.	__invoke()	Called when a script tries to call an object as a function.
13.	__set_state()	Called for classes exported by var_export() since PHP 5.1.0
14.	__clone()	Once the cloning is complete, if a _clone() method is defined, then the newly created object's _clone() method will be called, to allow any necessary properties that need to be changed.

Error Functions

These functions are used to handle errors and logging. The logging functions allow sending messages directly to other machines or systems while the error reporting functions allow customizing the level of error and feedback given. The behavior of the error functions are affected by the settings in the php.ini

1.	debug_backtrace()	Generates a backtrace
2.	debug_print_backtrace()	Prints a backtrace
3.	error_get_last()	Gets the last error occurred

4.	error_log()	Sends an error to a server error-log, to a file or to a remote destination.
5.	error_reporting()	Specifies which errors are reported.
6.	restore_error_handler()	Restores the previous error handler
7.	restore_exception_handler()	Restores the previous exception handler
8.	set_error_handler()	Sets a user defined function to handle errors.
9.	set_exception_handler()	Sets a user defined function to handle exceptions.
10.	trigger_error()	Creates a user-defined error message.
11.	user_error()	Alias of trigger error.

Error Constants

1.	E_ERROR1	Fatal run-time errors. Errors that cannot be recovered from. Execution of the script is halted
2.	E_WARNING2	Non-fatal run-time errors. Execution of the script is not halted
2.	E_PARSE4	Compile-time parse errors. Parse errors should only be generated by the parser
3.	E_NOTICE8	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
4.	E_CORE_ERROR16	Fatal errors at PHP startup. This is like an E_ERROR in the PHP core
5.	E_CORE_WARNING32	Non-fatal errors at PHP startup. This is like an E_WARNING in the PHP core
6.	E_COMPILE_ERROR64	Fatal compile-time errors. This is like an E_ERROR generated by the Zend Scripting Engine
7.	E_COMPILE_WARNING128	Non-fatal compile-time errors. This is like an E_WARNING generated by the Zend Scripting Engine
8.	E_USER_ERROR256	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
9.	E_USER_WARNING512	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
10.	E_USER_NOTICE1024	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
11.	E_STRICT2048	Run-time notices. PHP suggest changes to your code to help interoperability and compatibility of the code
12.	E_RECOVERABLE_ERROR4096	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
13.	E_ALL 6143	All errors and warnings, except of level E_STRICT

People are also reading: