Emezeta PHP Card Reference Cheat Sheet

PHP functions, parameters, data-type and others, by Manz

http://www.emezeta.com

Data type

mixed { boolean is_bool() | integer is_int() = is_long() | float = double is_float() = is_double() is_real() | string is_string() | array is_array() | object is_object() | resource is_resource() | NULL is_null() }

scalar is_scalar() { number is_numeric() }

floatval(var) = doubleval() float value

intval(var, base) integer value

strval(var) string value

get_defined_vars() show existent vars

gettype(var)

settype(var, type)

isset(var,var2...)

empty(var) var is empty?

unset(var,var2...) delete

print_r(expr, ret) show var

var_dump(expr,expr2...)

get_resource_type(res)

Comments

```
// comment #1
/* comment #2 */
# comment #3
```

function name
fixed parameter
optional parameter
description
constants or options
only in PHP5

Strings (slashes & quotes)

addcslashes(string,list_chars) add \ in chars of 'list_chars' in string

stripcslashes(string) remove \ in string processed with addcslashes

addslashes(string) add \ in chars in chars that need escape

stripslashes(string) remove \ in string processed with addslashes

htmlspecialchars(string,style,charset, doub_encode) convert special chars in HTML entities

htmlspecialchars_decode(string,style) revert htmlspecialchars efect

htmlentities(string,style,charset, doub_encode) convert ALL special chars in HTML entities

html_entity_decode(string,style,charset) revert htmlentities efect

get_html_translation_table(table,style) get translation table in htmlspecialchars & htmlentities

quotemeta(string) return a escape string of meta chars . \ + * ? [^] ($)

nl2br(string) convert newlines in <br /> break

strip_tags(string,exclude) erase HTML & PHP tags

specialchars
& &
" " (without ENT_NOQUOTES)
' ' (with ENT_QUOTES)
< <
> >

style
ENT_COMPAT (0) "
ENT_QUOTES (1) " &'
ENT_NOQUOTES (2)

charset
ISO-8859-1 Latin-1
ISO-8859-15 Latin-9 (€,...)
UTF-8 Unicode
cp866 DOS Cyrillic
cp1251 Win Cyrillic
cp1252 Win Europe
KOI8-R Russian
BIG5 Chinese (Taiwan)
GB2312 Chinese
BIG5-HKSCS Chinese (HongKong)
Shift_JIS Japanese
EUC-JP Japanese

Strings (conversions & crypt)

bin2hex(string) convert binary to hexadecimal

chr(ascii) return ASCII character

ord(string) return ASCII value of character

convert_cyr_string(string,from,to) convert from Cyrillic charset

convert_uuencode(string) uuencode a string

convert_uudecode(string) decode a uuencoded string

crc32(string) calculate cyclic redundancy checksum polynomial of string

crypt(string,salt) encrypt string with a encryption algorithm [0]

hebrev(string,max_chars_per_line) convert hebrew text to visual text

hebrevc(string,max_chars_per_line) = & convert newlines to
\n

localeconv() return localized numeric & monetary information

setlocale(category,locale...) set locale information

md5_file(file,raw_output) calculate md5 hash of a file

md5(string,raw_output) calculate md5 hash of a string

sha1_file(file,raw_output) calculate sha1 hash of a file

sha1(string,raw_output) calculate sha1 hash of a string

money_format(string,number) format a number as currency string

nl_langinfo(item) query language & locale information

number_format(number,dec,point,thousand) format number with:

salt
CRYPT_STD_DES (0) DES-based st.
CRYPT_EXT_DES (1) DES-based ext.
CRYPT_MD5 MD5 encryption
CRYPT_BLOWFISH Blowfish encrypt

raw_output
FALSE (0) return 32-chr (40 in sha1) hex number
TRUE (1) return 16-chr (20 in sha1) raw binary

(none) without decimals
dec number of decimals
point decimals separator
thousand thous. separator

Strings (splits)

chunk_split(string,length,end) split in strings of fixed 'length' [76] & put end string [\n\r].

str_split(string,length) split in array of strings of fixed 'length' [76]

explode(del,string,limit) split in array of strings separated by delimiter (max. 'limit' elements)

implode(del,array) join in a string separated by 'del' all elements of a array = join()

strtok(string,token) split a string into a smaller substrings (tokens)

Strings (string & chars operations)

count_chars(string,mode) count number of ocurrences of every char in string

levenshtein(str1,str2,cost_ins,cost_rep,cost_del) calculate distance between 2 strings

similar_text(str1,str2,percent) calc similarity between 2 strings

soundex(string) calculate words pronounced similarly (soundex key)

metaphone(string,phones) = but is more accurate than soundex()

echo(string,string2...) output strings = <?="string"?>

ltrim(string,charlist) erase whitespaces & 'charlist' from string left

rtrim(string,charlist) = but from right of string = chop()

trim(string,charlist) = ltrim() + rtrim()

str_repeat(string,times) repeat string x 'times'

str_rot13(string) return ROT13 version of string

str_shuffle(string) randomize all chars in a string

strpos(string,substr,offset) return pos of first ocurrence

stripos(string,substr,offset) = but case-insensitive

strrpos(string,char,offset) return pos of last ocurrence (substr in php5)

strripos(string,char,offset) = but case-insensitive

strrchr(string,char) return substr from last-pos to end

strstr(string,substr) return substr from first-pos to end = strchr()

stristr(string,substr) = but case-insensitive

str_replace(search,replace,string,times) replace substring 'search' => 'replace' & return times

str_ireplace(search,replace,string,times) = but case insensitive

str_pad(string,length,padstr,padtype) fill string to 'length' with 'padstr'

str_word_count(string,format,charlist) count words in a string

strcspn(string,charlist,begin,end) return length of substring which not contain any char

strspn(string,charlist,begin,end) return length of substring which contain any char

strpbrk(string,charlist) return substr from pos of first ocurrence of a char in 'charlist'

strlen(string) return length of string

strrev(string) return a reverse string

strtolower(string) return string with all alphabetic chars in lowercase

strtoupper(string) return string with all alphabetic chars in uppercase

ucfirst(string) return string with the first character of str capitalized (if is alphabetic)

ucwords(string) return string with the first character of each word capitalized (if is alphabetic)

strtr(string,from,to) translate chars from => to (or pair string) array) in string

substr(string,begin,length) return a substring of string (with a optional length)

substr_replace(string,substr,begin,length) insert or replace substr in string from begin-pos

substr_count(string,substr,begin,length) count number times of substring ocurrences

wordwrap(string,width,break,cut) wrap string in a lines of 'width' [75] length with break [\n]

mode
0 array with ocurrences of all chars
1 = (0 value exclude)
2 = (0 value only)
3 string with all chars used
4 string with all chars unused

whitespaces
" " ASCII 32 (0x20), space
\t ASCII 9 (0x09), tab
\n ASCII 10 (0x0A), new line
\r ASCII 13 (0x0D), carriage return
\0 ASCII 0 (0x00), NUL-byte
\x0B ASCII 11 (0x0B), vertical-tab

padtype
STR_PAD_RIGHT (0) STRING_
STR_PAD_LEFT (1) _STRING
STR_PAD_BOTH (2) _STRING_

Strings (compare)

strcmp(str1,str2) a binary safe string compare

strcasecmp(str1,str2) = but case-insensitive

strnatcmp(str1,str2) = strcmp() but natural order

strnatcasecmp(str1,str2) = but case-insensitive

strncmp(str1,str2,len) = strcmp() but only compare 'len' first chars

strncasecmp(str1,str2,len) = but case-insensitive

strcoll(str1,str2) locale string compare

substr_compare(str1,str2,offset,length,case) compare str1 from 'offset' pos to 'length' pos

<0 str1 less str2
=0 str1 equal str2
>0 str1 great str2

Strings (print family)

print(string) output a string

printf(format,args...) output a formatted string

vprintf(format,args...) = but accepts an array of arguments

sprintf(format,args...) return a formatted string

vsprintf(format,args...) = but accepts an array of arguments

sscanf(string,format,args...) parses input from a formatted string

fprintf(handle,format,args...) write a formatted string to a stream

vfprintf(handle,format,args...) = but accepts an array of arguments

fscanf(handle,format,args...) parses input from a formatted file

format
0 return number of words found
1 return array with words found
2 return array with pos & words

Regular Expressions

POSIX Extended RegExp

ereg(regexp,string,reg_group) search for match regexp pattern in string (& group parenthesized)

eregi(regexp,string,reg_group) = but case-insensitive

ereg_replace(regexp,replace,string) search regexp & replace match text with 'replace'

eregi_replace(regexp,replace,string) = but case-insensitive

split(regexp,string,limit) split string into array by a regexp

spliti(regexp,string,limit) = but case-insensitive

sql_regcase(string) create a regexp for case insensitive match

grep_flags
PREG_GREP_INVERT not match

Perl Compatible RegExp

preg_grep(regexp,input,grep_flags) return array entries that match the pattern regexp

preg_match(regexp,string,match,flags,offset) perform regexp match

preg_match_all(regexp,string,match,flags,offset) = but if found, continue search

preg_quote(string,del) add slash regexp chars . \ + * ? [^] $ ( ) { } = ! < > | :

preg_replace(regexp,replace,string,limit,times) perform a regexp search & replace

preg_replace_callback(regexp,callback,string,limit,times) = but using a callback func.

preg_split(regexp,string,limit,flags) split string by a regular expression

preg_last_error() return error code of last Perl Compatible RegExp

flags
PREG_PATTERN_ORDER (0)
PREG_SET_ORDER (1)
PREG_OFFSET_CAPTURE (2)

sflags
PREG_SPLIT_NO_EMPTY (0)
PREG_SPLIT_DELIM_CAPTURE (1)
PREG_SPLIT_OFFSET_CAPTURE (2)

regular expression
\ escape
^ start of subject
$ end of subject
. any character (except newline)
| start of alternative branch
[start char class definition
] end char class definition
in square brackets...
^ negate (if is first char)
- char range indicate
(start subpattern
) end subpattern
? quantifier minimizer (0 or 1)
* 0 or more quantifier
+ 1 or more quantifier
{ start min/max quantifier
} end min/max quantifier

regular expression (2)
\d any decimal digit
\D any char not a decimal digit
\h horizontal whitespace char
\H not horizontal whitespace...
\s any whitespace char
\S any not whitespace char
\v any vertical whitespace char
\V any not vertical whitespace...
\w any word character
\W any "non-word" character
\n newline
\r carriage return
\t tab

Operators

Arithmetics [+] [-] [*] [/] [%]

Assignments [=] [+=] [-=] [*=] [/=]

Bitwise [&] [|] [^] [~] [<<] [>>]

Comparison [==] [===] [!=] [<>] [!==] [<] [>] [<=] [>=]

Error control [@]

Execution [`]

Increase/Decrease [++var] [var++] [--var] [var--]

Logical [and, &&] [or, | |] [xor] [!var]

String [.] [.=]

Array [+] [==] [===] [!=, <>] [!===]

Type [instanceof]

Constants

define(name,value, sensitive)

constant(name) show const value

defined(name) check if is defined

get_defined_constants(categ.)

Functions

function name_function(var, var2 = "default"...) {...}

function_exists(function) check if function exists

get_defined_functions() show defined functions

call_user_func(func,param1...) exec. function

is_callable(var, syntax, name) check

Math functions

abs(number) absolute value

exp(arg) calc. exponent of e

fmod(x,y) get modulo of division

log10(arg) base-10 logarithm

log(arg,base) natural logarithm

max(values) find highest value

min(values) find lowest value

pi() get value of pi

pow(base,exp) exponent. expr.

sqrt(arg) square root

is_finite(value) check finite value

is_infinite(value) check infinite val.

is_nan(value) check Not a Number

base_convert(num,from,to) convert number between bases

decbin(dec) convert to binary

bindec(bin) convert to decimal

dechex(dec) convert to hexadec.

hexdec(hex) convert to decimal

decoct(dec) convert to octal

octdec(oct) convert to decimal

deg2rad(deg) convert to radian

rad2deg(rad) convert to degree

getrandmax() show max rand value

mt_getrandmax() = but mt algorith.

rand(min,max) create rand int.

mt_rand(min,max) = but mt alg.

srand(seed) seed rand generator

mt_srand(seed) = but mt algorith.

lcg_value() generate pseudo-rand

ceil(value) round fractions up

floor(value) round fractions down

round(value,pr) round 'pr' decim.

sin(arg) sine

sinh(arg) hyperbolic sine

asin(arg) arc sine

asinh(arg) inv. hyperbolic sine

cos(arg) cosine

cosh(arg) hyperbolic cosine

acos(arg) arc cosine

acosh(arg) inv. hyperbolic cosine

tan(arg) tangent

tanh(arg) hyperbolic tangent

atan(arg) arc tangent

atanh(arg) inv. hyperbolic tangent

atan2(x,y) arc tangent of 2 var

hypot(x,y) calc. hypotenuse

Character type

Check if only are present in text...

ctype_alnum(text) alphanumeric chars

ctype_alpha(text) alphabetic chars

ctype_cntrl(text) control chars (tab,esc...)

ctype_digit(text) digits

ctype_graph(text) visible chars & space

ctype_lower(text) lowercase chars

ctype_upper(text) uppercase chars

ctype_print(text) printable chars

ctype_punct(text) = except space & alphanumeric

ctype_space(text) blank spaces (tab, space, \n...)

ctype_xdigit(text) hexadecimal digit

Predefined vars

$GLOBALS
$_GET
$_POST
$_COOKIE
$_REQUEST
$_FILES
$_ENV
$_SESSION
$_SERVER

PHP_SELF running script
GATEWAY_INTERFACE CGI specification
SERVER_ADDR Server IP address
SERVER_NAME Server Hostname
SERVER_SOFTWARE Server name & version
SERVER_PROTOCOL Server protocol
SERVER_ADMIN Admin mail address
SERVER_PORT Server Port
SERVER_SIGNATURE Version and Virtual Host
REQUEST_METHOD { GET/POST/PUT/HEAD }
REQUEST_TIME timestamp init
QUERY_STRING GET params
DOCUMENT_ROOT main dir
HTTP_ACCEPT MIME type
HTTP_ACCEPT_CHARSET Charset
HTTP_ACCEPT_ENCODING Compress
HTTP_ACCEPT_LANGUAGE Language
HTTP_CONNECTION Persistent connection { Keep-alive/close }
HTTP_HOST Main domain server (no subdomain)
HTTP_REFERER Client back webpage
HTTP_USER_AGENT Client browser (header)
HTTPS Secure HTTP protocol
REMOTE_ADDR Client IP
REMOTE_HOST Client Hostname
REMOTE_PORT Client Port
SCRIPT_FILENAME Running server script actually
PATH_TRANSLATED = SCRIPT_FILENAME
SCRIPT_NAME Script path (relative to root)
REQUEST_URI URI from request (similar to SCRIPT_NAME)
PHP_AUTH_DIGEST HTTP Digest Header (with Apache as module)
PHP_AUTH_USER Username in authorization
PHP_AUTH_PW Password in authorization
AUTH_TYPE Authorization type
argc Number of params
argv Array with GET params

Server data

Client header

Client data

Date & time

checkdate(month,day,year) check if date is correct (leap inclusive)

getdate(timestamp) get date/time info

time() return current timestamp

microtime(float) return current timestamp (with microseconds)

date_default_timezone_get() return default timezone used

date_default_timezone_set(timezoneid) set a default timezone

date_parse(strtotime) return array with (day, month, year...) info

gettimeofday(float) call 'gettimeofday' command, return array or float

gmdate(date,timestamp) format a GMT/UTC date/time

date(date,timestamp) format a local time/date as integer

strptime(strtime,date) parse a strftime() time/date

strtotime(sengdate,now) parse english datetime in timestamp

gmmktime(hour,min,secs,month,day,year,is_dst) get a timestamp GMT

gmstrftime(cformat,timestamp) format a GTM/UTC time/date (with locale sets)

strftime(cformat,timestamp) format a local time/date (with locale sets)

localtime(timestamp,key_str) get local time (false key_str, numeric array keys)

mktime(hour,min,secs,month,day,year,is_dst) get timestamp for a date

date_sun_info(timestamp,lat,long) sunset/sunrise & twilight begin/end

date_sunrise(timestamp,format,lat,long,zenit,gmt) get sunrise time

date_sunset(timestamp,format,lat,long,zenit,gmt) get sunset time

format
SUNFUNCS_RET_STRING (0)
SUNFUNCS_RET_DOUBLE (1)
SUNFUNCS_RET_TIMESTAMP (2)

Date parameters

date(date,timestamp) format a local time (or a timestamp time)

DAY d 08 D Mon j 8 l (L) Monday N 1...7 (week day) S st, th... w 0...6 (week day) z 0...365
WEEK W 1...52 (week of year)
MONTH F January m 01 M Jan n 1 t 28...31 (month days)
YEAR L 1 (leap year) 0 (no) o 2008 Y 2008 (W) y 08
TIME a am A AM B 000...999 (swatch internet time)
HOUR g 6 G 18 h 06 H 18
MIN i 02 (min) s 02 (sec) u 54321 (millisec)
TIMEZONE e UTC, GMT,...(id) I (i) 1 (daylight saving) 0 (no) O +0200 (GMT diff hours)
P +02:00 (= GMT diff hr & min) T EST, MDT (abbr) Z 50400 (offset)
FULL c 2004-02-12T15:19:21+00:00
r Thu, 21 Dec 2000 16:01:07 +0200
U UNIX Epoch [see time()]

DateTime & DateTimeZone class

date_create(strtotime,DTzone) create a DateTime = constructor

date_date_set(datetime,year,month,day) set a date in DT object = setDate()

date_format(datetime,fdate) return date with specified format = format()

date_isodate_set(datetime,year,week,day) set ISO date = setISODate()

date_modify(datetime,strtotime) alter a DateTime object = modify()

date_offset_get(datetime) return daylight saving time offset = getOffset()

date_time_set(datetime,hour,min,secs) set time = setTime()

date_timezone_get(datetime) return timezone of date = getTimezone()

date_timezone_set(datetime,DTzone) set a date timezone = setTimezone()

timezone_abbreviations_list() return (dst, offset, name) = listAbbreviations()

timezone_identifiers_list() return numeric Index with all ids = listIdentifiers()

timezone_name_get(DTzone) return name of timezone = getName()

timezone_offset_get(DTzone,datetime) return timezone offset from GMT = getOffset()

timezone_open(timezoneID) return DateTimeZone object (DTzone) = constructor

timezone_transitions_get(DTzone) return all transitions for timezone = getTransitions()

timezone_name_from_abbr(abbr,gmtOffset,is_dst) return name from abbrev.

Filesystem (basic functions)

basename(path,suffix) return filename of a path filename

chgrp(file,group) change file group = lchgrp() for link file

chmod(file,mode) change file mode

chown(file,user) change file owner = lchown() for link file

copy(src,dst,context) copy a file

disk_free_space(dir) return available space in disk of dir = diskfreespace()

disk_total_space(dir) return total size of disk of dir

file_exists(file) check if file or dir exists

file_get_contents(file,flags,context,offset,maxlen) get content file into string

file_put_contents(file,data,flags,context) put content string into a file

file(file,flags,context) get content file into array

filesize(file) attempts create dir

rename(old,new,context) rename a file or dir

stat(file) give information about a file

tempnam(dir,preffix) create unique filename

fnmatch(wildcard,str,flags) match filenames

glob(wildcard,flags) find pathnames match wildc.

link(target,link) create a hard link

symlink(target,link) create a symbolic link

linkinfo(path) get information about a link

readlink(path) return target of a symbolic link

realpath(path) return absolute real path of link

move_uploaded_file(file,dst) move upload file

parse_ini_file(file,sections) parse a .ini file

pathinfo(file,options) return info about file path

readfile(file,use_inc,context) outputs a file

touch(file,time,atime) uptime time of file

umask(mask) change the current umask

unlink(file,context) delete a file

fileatime(file) get last access

filemtime(file) get last modified

filegroup(file) get group

fileinode(file) get inode

fileowner(file) get owner

fileperms(file) get permission

filetype(file) get type (fifo, dir...)

is_dir(file) check if is a directory

is_executable(file) = exec. file

is_file(file) check if is a reg. file

is_link(file) check if is a link

is_readable(file) = read file

is_uploaded_file(file) = upload

is_writable(file) = is_writeable

options
(default) (0) return all elements
PATHINFO_DIRNAME (1)
PATHINFO_BASENAME (2)
PATHINFO_EXTENSION (3)
PATHINFO_FILENAME (4)

Control Structures

```
if (expr)
  statement #1;
elseif (expr)
  statement #2;
else
  statement #3;

((expr)? expr_true : expr_false )

for (expr_init; expr; expr_incr)
  statement #1;

foreach ( expr as key => value)
  statement #1;

while (expr)
  statement #1;

do {
  statement #1;
} while (expr);

switch (expr) {
  case 1:
    echo "expr equals 1";
    break;
  ...
  default:
    echo "default option";
    break;
}

continue stop iteration & go next
break stop iteration & finish loop
return expr end & return param
```

Include files

require(file) include PHP script file (error_fatal if don't exist)

include(file) include PHP script file (warning if don't exist)

require_once(file) = require, but first check if file was included

include_once(file) = include, but first check if file was included

set_include_path(path) set new include path

get_include_path() return include path

restore_include_path() set default include path

get_included_files() return all files included

URL functions

base64_encode(data) encode data with MIME base64

base64_decode(data,strict) decode data encoded with MIME base64

parse_str(string,array) create array with keys & values of URL query string

get_headers(url,text_keys) fetch all HTTP headers of a URL request

get_meta_tags(file,use_include_path) extract all meta tags in array

http_build_query(formdata,num_pref,arg_sep) create URLencode query

parse_url(url,component) parse URL and return its components

rawurlencode(string) encode string into raw url (%20, %40...)

rawurldecode(string) decode raw url into string

urlencode(string) URL-encode string

urldecode(string) URL-decode string

Directory functions

chdir(dir) change current directory

chroot(dir) change current root directory

dir(dir) return an instance of Directory class (read, rewind & close)

opendir(path,context) open a directory handle

getcwd() get current work directory

readdir(handle) read entry directory handle

rewinddir(handle) rewind directory handle

closedir(handle) close directory handle

scandir(dir,sort_order,context) list files & dir inside specified dir

dirname(path) return dir path of a file path string

is_dir(file) check if a file is a directory

mkdir(pathname,chmod,recursive,context) attempts create dir

rmdir(pathname,context) attempts remove empty dir

Miscellaneous functions

connection_aborted() check whether client disconnected

connection_status() return connection status: (0) NORMAL (1) ABORTED (2) TIMEOUT

exit(status) output message (or return integer value) & finish script = die()

eval(code_str) evaluate string as PHP code

highlight_file(file,return) syntax highlighting of a PHP file = show_source()

highlight_string(string,return) syntax highlighting of a PHP string

php_strip_whitespace(file) return code without comments & whitespaces

usleep(msecs) delay script execution for 'msecs' micro-seconds

sleep(secs) delay script execution for 'secs' seconds

time_nanosleep(secs,nanoseconds) = but delay seconds & 'nanosecs' too

time_sleep_until(timestamp) delay script execution until 'timestamp' time

sys_getloadavg() return average system load (UNIX 'uptime' command)

Execution functions

exec(command,output,return) execute an external program

passthru(command,return) = and display RAW output

shell_exec(command) execute command via shell & return output = ``

system(command,return) execute an external program & display output

Arrays

matrix[key] = value; one-dimension array

matrix[key][key] = value; two-dimension array

array(elements) create a array

case
CASE_LOWER (0)
CASE_UPPER (1)

array_change_key_case(mat,case) change all key to up/low case

array_chunk(mat,size,savekeys) split in arrays of 'size' elements

array_combine(mat,values) create array with 2 arrays (keys & values)

array_count_values(mat) return array with # of frequency of values

array_fill_keys(mat,value) create array with 'mat' keys and same value

array_fill(init,num,value) = but since 'init' to 'init+'num' index

array_filter(mat,func) return mat with values that return true in function

array_flip(mat) return mat swap keys & values

array_key_exists(key,mat) check if key exists in 'mat'

array_keys(mat,value,strict) return all keys of array

array_map(func,mat,mat2...) process & return all values with function

array_merge(mat,mat2...) merge two or more arrays

array_merge_recursive(mat,mat2...) = but with recursive mode

array_pad(mat,size,fix) create arr same 'mat' & fixed with 'fix'

array_product(mat) calculate product value in elements of arr.

array_rand(mat,num) select >1 (or more) random elements

array_reduce(mat,func,init) apply func to obtain one element

array_search(value,mat,strict) find value and return key if found

array_slice(mat,offset,size,savekeys) extract a subarray

array_splice(mat,offset,size,newmat) replace a subarray

array_sum(mat) sum all elements in array

array_unique(mat) remove duplicated values in array

array_walk(mat,func,params) apply func. to every elements of array

array_walk_recursive(mat,func,params) = but in recursive mode

compact(var...) create array with key (var's name) & value (var's content)

in_array(value,mat,strict) check if a value exists in array

list(var...) assign vars such as array

range(init,end,incr) create array with number range (similar to for)

mode
nothing (0)
COUNT_RECURSIVE (1)

Arrays basic movements

array_pop(mat) extract (pop) last element from array

array_push(mat,var...) insert (push) 1 or more elements in end of array

array_shift(mat) extract (shift) first element from array

array_unshift(mat,var...) insert 1 or more elements at init of array

count(var,mode) count elements in array = sizeof()

current(mat) return current element in array = pos()

each(mat) return current (key,value) pair and go next

end(mat) go last element in array and return it

key(mat) return key of current element

next(mat) go next element in array and return it

prev(mat) go prev element in array and return it

reset(mat) go first element in array and return it

flags
SORT_ASC (0)
SORT_DESC (1)

sort
SORT_REGULAR (0)
SORT_NUMERIC (1)
SORT_STRING (2)
SORT_LOCALE_STRING (3)

Arrays sort methods

array_multisort(mat,sort,mat2...) sort several arr. or multidim. arr

array_reverse(mat,savekeys) array with elements in rev. order

asort(mat,flags) sort array in order, maintain index

arsort(mat,flags) sort array in reverse order, maintain index

ksort(mat,flags) sort array (for key) in order (maintain assoc)

krsort(mat,flags) sort array (for key) in reverse order (maintain assoc)

natsort(mat) sort in natural order (maintain assoc)

natcasesort(mat) sort in natural order (maintain assoc) & no case sensitive

rsort(mat,flags) sort array in reverse order (max to min)

shuffle(mat) randomize elements in array

sort(mat,flags) sort array (min to max)

usort(mat,func) sort array with custom func.

uksort(mat,func) sort array with custom func. (respect keys)

uasort(mat,func) sort array with custom func. (maintain assoc)

Arrays as sets (difference & intersect)

array_diff(mat,mat2,mat3...) difference between mat

array_intersect(mat,mat2,mat3...) intersect between mat

array_?_key(mat,mat2,mat3...) ...between mat (key sensitive)

array_?_ukey(mat,mat2,mat3...,func) = but with custom func.

array_udiff(mat,mat2,mat3...,func) diff with custom func

array_uintersect(mat,mat2,mat3...,func) intersect with custom func.

array_?_assoc(mat,mat2,mat3...) diff between mat (index sensitive)

array_?_uassoc(mat,mat2,mat3...,func) = but with custom func.