Data type

mixed
- boolean is_bool()
- integer is_int() = is_integer() 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()

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)

floatval(var) = doubleval() float value
intval(var, base) integer value
strval(var) string value
get_defined_vars() show existent vars

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

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

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

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

Strings (print family)

1 print(string) output a string
2 printf(format, args...) output a formatted string
2 vprintf(format, args...) = but accepts an array of arguments
sprintf(format, args...) return a formatted string

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,2style,charset, doub_encode) convert special chars in HTML entities
htmlspecialchars_decode(string,2style) revert htmlspecialchars efect
htmlentities(string,2style,charset, doub_encode) convert ALL special chars in HTML entities
html_entity_decode(string,2style,charset) revert htmlentities efect
get_html_translation_table(2table,2style) 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) " & '

charset
ISO-8859-1 Latin-1
ISO-8859-15 Latin-9 (€,...)

Strings (splits)

chunk_split(string,2length,end) split in strings of fixed 'length' [76] & put end string [\n\r].
str_split(string,2length) split in array of strings of fixed 'length' [76]
explode(del,string,2limit) 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,2mode) count number of ocurrences of every char in string
2levenshtein(str1,str2,2cost_ins,2cost_rep,2cost_del) calculate distance between 2 strings
2similar_text(str1,str2,2percent) calc similarity between 2 strings
soundex(string) calculate words pronounced similarly (soundex key)
metaphone(string,2phones) = 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,2times) repeat string x 'times'
str_rot13(string) return ROT13 version of string
str_shuffle(string) randomize all chars in a string
2strpos(string,substr,2offset) return pos of first ocurrence
2stripos(string,substr,2offset) = but case-insensitive
2strrpos(string,char,2offset) return pos of last ocurrence (substr in php5)
2strripos(string,char,2offset) = 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,2times) replace substring 'search' => 'replace' & return times
str_ireplace(search,replace,string,2times) = but case insensitive
str_pad(string,2length,padstr,2padtype) fill string to 'length' with 'padstr'
str_word_count(string,2format,charlist) count words in a string
2strcspn(string,charlist,2begin,2end) return length of substring which not contain any char
2strspn(string,charlist,2begin,2end) return length of substring which contain any char
strpbrk(string,charlist) return substr from pos of first ocurrence of a char in 'charlist'
2strlen(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,2begin,2length) return a substring of string (with a optional length)
substr_replace(string,substr,2begin,2length) insert or replace substr in string from begin-pos
substr_count(string,substr,2begin,2length) count number times of substring ocurrences
wordwrap(string,2width,break,cut) wrap string in a lines of 'width' [75] length with break [\n]
2strcmp(str1,str2) a binary safe string compare

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_

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

Arrays

```
matrix[ key] = value;  one-dimension array
matrix[ key][ key] = value;  two-dimension array
array(elements)  create a array

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)
```

case
CASE_LOWER (0)
CASE_UPPER (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
```

mode
nothing (0)
COUNT_RECURSIVE (1)

Exceptions	
try **{** **// Code that may cause an Exception or Error.** **}** **catch (Throwable $t)** **{** **// Executed only in PHP 7, will not match in PHP 5** **}** **catch (Exception $e)** **{** **// Executed only in PHP 5, will not be reached in PHP 7** **}**	**getMessage(); returns the name of the exception** **getCode(); returns the code of the exception** **getFile(); returns the name of the file where the exception happened** **getLine(); returns the line in the file where the exception happened** **__toString(); returns the exception in a String format**

MYSQLi OO

$db = new mysqli($servername, $username, $password, $dbname); Create the connection
$db→close(); Close the connection
$result=$conn->multi_query($sql)): submit a multiquery
$result = $conn->query($query); Submit a query

Methods with Object 'result' from a query:
fetch_array() method: creates an associative array, a numeric
array, or both, based on the second parameter (MYSQLI_ASSOC, MYSQLI_NUM, or MYSQLI_BOTH).
fetch_assoc() method: creates an associative array, using the
data field names as the array keys.
fetch_row() method: creates a numeric array, using numeric indexes for each data field (starting at 0, and using the data field order specified in the table or SELECT statement data field).
close(): It is not necessary to free memory because php frees it at the end of the script. But in cases of high traffic it can be useful
num_rows: number of rows

Passwords

$strongpass = password_hash($pass, PASSWORD_DEFAULT);
password_verify($pass1, $pass2)

MYSQLi PROCEDURAL

$conn = mysqli_connect(host, user, password, database, port, socket);
mysqli_close($conn);
$result = mysqli_query($conn,$sql);
$num= mysqli_num_rows($result)
$fila = mysqli_fetch_array($result);

prepared statement:
$sql = "INSERT INTO emp VALUES (?, ?, ?, ?, ?, ?)";
Then you use the prepare() method to submit it:
$stmt = $conn->prepare($sql);
And bind_param to pass the values.
$stmt->bind_param("isssi", $empid, $lname, $fname, $start, $birth, $salary);
The first parameter defines the data type of each of the data Values:
b: A blob data type value
i: An integer data type value
d: A double data type value
s: A string data type value
And finally, you must execute the prepared statement:
$stmt->execute();

PDO

$pdo = new PDO("mysql:host=$servername;dbname=$dbname",$username, $password);
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$pdo->exec('SET NAMES "utf8"');

$affectedRows=$pdo→exec($sql) execute the query DELETE, INSERT, and UPDATE
$result=$pdo->query($sql)For SELECT queries

$row = $result->fetch() returns the next row

prepared statement: With variables

$sql = 'INSERT INTO comment values (:commenttext)';
$stmt = $pdo->prepare($sql);

prepare statements
$stmt->bindValue(':commenttext', $commenttext);
$stmt->execute();

Or more concise
$stmt->execute([':commenttext' => $commenttext]);

prepared statement: with question mark

$sql = 'INSERT INTO comment values (?,?)';
$stmt = $dbh->prepare($sql);

$stmt->bindParam(1, $comment);
$stmt->bindParam(2, $date);
$stmt->execute();

Transactions:

$pdo->beginTransaction();
$pdo->commit();//save changes

$pdo->rollBack(); //undo changes

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

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
idate(date, timestamp) format a local time/date as integer
strptime(strftime, 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, cenit, gmt) get sunrise time
date_sunset(timestamp, format, lat, long, cenit, gmt) get sunset time

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.

COOKIES
setcookie(name, value, expire, path, domain, secure, httponly);