Unit-4

PHP and XML 8

INTRODUCTION TO PHP

The PHP Hypertext Pre-processor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases

PHP is basically used for developing web based software applications.PHP is probably the most popular scripting language on the web. It is used to enhance web pages.PHP is known as a **server-sided language.** That is because the PHP doesn't get executed on the client's computer, but on the computer the user had requested the page from. The results are then handed over to client, and then displayed in the browser.

Features of PHP:

- PHP is a server side scripting language that is embedded in HTML
- PHP was originally developed by the **Danish Greenlander Rasmus Lerdorf**, and was subsequently developed as open source.
- It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP supports a large number of protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP language tries to be as forgiving as possible.
- PHP syntax is C-Like.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.

- ❖ PHP can handle forms, i.e. gather data from files, save data to a file, through email the user can send data, return data to the user.
- ❖ The user can add, delete, and modify elements within the database through PHP.
- ❖ They can access cookies variables and set cookies.
- ❖ Using PHP, the user can restrict users to access some pages of the website.
- ❖ It can encrypt data.

Working of PHP

When the client requests a PHP page residing on the server, the server first performs the operations mentioned by the PHP code of the page. Then is sends the output of the PHP page in HTML format. So when the user views the source code of the page, it will be full of HTML tags. All the work is done at the server side.

4.1 PROGRAMMING WITH PHP

➢ **Comments**

A comment is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP:

- **Single-line comments:** They are generally used for short explanations or notes relevant to the local code.
- **Multi-line comments:** They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C.

Rules of PHP

- ❖ PHP is white space insensitive
- ❖ PHP is case sensitive
- ❖ Statements are expressions terminated by semicolons
- ❖ Expressions are combinations of tokens
- ❖ Braces make blocks

➢ **PHP Variable Types**

All variables in PHP are denoted with a leading dollar sign ($). The value of a variable is the value of its most recent assignment. Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.

Variables in PHP do not have **intrinsic types (data types)** - a variable does not know in advance whether it will be used to store a number or a string of characters.

Variables used before they are assigned have default values. PHP automatically converts types from one to another when necessary. PHP has a total of eight data types:

- Integers are whole numbers, without a decimal point. They can be in decimal, octal or hexadecimal. Eg: 87.
- Doubles are floating-point numbers. Eg: 3.87
- Booleans have only two possible values either true or false.
- NULL is a special type that only has one value: NULL
- Strings are sequences of characters
- Arrays are named and indexed collections of other values.
- Objects are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- Resources are special variables that hold references to resources external to PHP (such as database connections).
- The first five are simple types, and the arrays and objects are compound types.
- The compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types: Local variables, Function parameters, Global variables and Static variables.

Variable Naming

Rules for naming a variable are:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but the user cannot use characters like + , - , % , (,) . & , etc

PHP Constants

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. To define a constant ,

use define() function and retrieve the value of a constant. The function constant() is used to read a constant's value .

define(name, value, case-insensitive)

The name specifies the name of the constant, value: Specifies the value of the constant and case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

Differences between constants and variables in PHP

Constants in PHP	Variables in PHP
No $ sign before constants.	$ sign is present before variables.
Constants are defined using define().	Variables are defined using assignment statement.
Constants may be defined and accessed anywhere without any regard.	Variables follow certain scope.
Constants cannot be redefined or undefined.	They can be redefined.

Pre-defined constants

Name	Description
__LINE___	The current line number of the file.
__FILE___	The full path and filename of the file. If used inside an include, the name of the included file is returned. Since PHP 4.0.2, __FILE__always contains an absolute path whereas in older versions it contained relative path under some circumstances
__FUNCTION___	The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__CLASS___	The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__METHOD___	The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive).

- **Echo and Print statements**

Differences between echo and print statement

echo	print
This does not have return value.	This has a return value of 1.
This cannot be used in expressions.	This can be used in expressions.
This can take multiple parameters.	This can take only one parameter.
This is slightly faster than print.	This is comparatively slower.

- **Operators**

Operators are used to perform operations on variables and values.PHP divides the operators in the following groups: Arithmetic operators, Assignment operators, Comparison operators, Increment/Decrement operators, Logical operators, String operators and Array operators.

PHP CONTROL STATMENTS

Decision Making Statements

The if, else if ...else and switch statements are used to take decision based on the different condition.

- if statement - executes some code only if a specified condition is true.
- if...else statement - executes some code if a condition is true and another code if the condition is false.
- if...else if ...else statement - specifies a new condition to test, if the first condition is false
- switch statement - selects one of many blocks of code to be executed

- **if statement**

if statement

```
<?php
$t = date("H");
if ($t < "20") //This program outputs the echo statement if the hour is <20
{
   echo "Have a good day!";
}
?>
```

- **if-else statement**

  Use the if ...else statement to execute some code if a condition is true and another code if the condition is false.

if else statement

```
<?php
$t = date("H");
if ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
}
?>
```

- **if-else ladder**

  Use the if....else if...else statement to specify a new condition to test, if the first condition is false.

if-else ladder

```
<?php
$t = date("H");
if ($t < "10") {
   echo "Have a good morning!";
} elseif ($t < "20") {
   echo "Have a good day!";
} else {
   echo "Have a good night!";
}
?>
```

- **switch statement**

  To select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code. The switch statement works in an unusual way. First it evaluates given expression then seeks a label to match the resulting value. If a matching value is found then the code associated with the matching

label will be executed or if none of the labels match then statement will execute any specified default code.

Switch statement

```
<html>
<body>
<?php $d=date("D");
switch ($d)
{
case "Mon":
  echo "Today is Monday";
  break;
case "Tue":
  echo "Today is Tuesday";
   break;
case "Wed":
  echo "Today is Wednesday";
   break;
case "Thu":
   echo "Today is Thursday";
    break;
case "Fri":
     echo "Today is Friday";
     break;
case "Sat":
    echo "Today is Saturday";
    break;
case "Sun":
    echo "Today is Sunday";
   break;
default:
    echo "Wonder which day is this ?";
```

```
}
?>
</body></html>
```

Looping Statements

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types

- for : loops through a block of code a specified number of times.
- while: loops through a block of code if and as long as a specified condition is true.
- do...while: loops through a block of code once, and then repeats the loop as long as a special condition is true.
- foreach: loops through a block of code for each element in an array.

➢ **For loop**

for loop

```
<html>
<body>
<?php
$a = 0;
$b = 0;
for( $i=0; $i<5; $i++ )
{
 $a += 10;
$b += 5;
}
echo ("At the end of the loop a=$a and b=$b" );
?>
</body></html>
```

```
At the end of the loop a=50 and b=25
```

➢ **While loop**

The while statement will execute a block of code as long as a test expression is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

While loop

```
<html> <body>
<?php
$i = 0;
$num = 50;
while( $i < 10)
{
$num--;
$i++;
}
echo ("Loop stopped at i = $i and num = $num" );
?> </body></html>
```

Loop stopped at i = 1 and num = 40

- **Do…while loop**

```
<html> <body>
<?php
$i = 0;
$num = 0;
do {
    $i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i" );
?></body></html>
```

Loop stopped at i = 10

- **For Each loop**

The for each statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed**.**

```
<html> <body>
<?
php $array = array( 11, 12, 13,14, 15);
```

```
foreach( $array as $value )
{
  echo "Value is $value <br />";
} ?> </body></html>
```

```
Value is 11
Value is 12
Value is 13
Value is 14
Value is 15
```

- **Break statement**

  The PHP break keyword is used to terminate the execution of a loop prematurely. The break statement is situated inside the statement block. After coming out of a loop immediate statement to the loop will be executed.

- **Continue statement**

  The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop. Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

FUNCTIONS

A function is a block of statements that can be used repeatedly in a program. A function will not execute immediately when a page loads. A function will be executed by a call to the function. There are two types of functions: Built-in functions and User defined functions

User defined Functions

A user defined function declaration starts with the word function. Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses.

```
<?php
function sum($x, $y) {
   $z = $x + $y;
   return $z; }
echo "5 + 10 = " . sum(5, 10) . "<br>";
```

```
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4); ?>
```

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

Built-in functions

Array functions

Function	Description
array()	Creates an array
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
array_fill()	Fills an array with values
array_fill_keys()	Fills an array with values, specifying keys
array_filter()	Filters the values of an array using a callback function
array_flip()	Flips/Exchanges all keys with their associated values in an array
array_intersect()	Compare arrays, and returns the matches (compare values only)
array_key_exists()	Checks if the specified key exists in the array
array_keys()	Returns all the keys of an array
array_map()	Sends each value of an array to a user-made function, which returns new values
array_merge()	Merges one or more arrays into one array
array_multisort()	Sorts multiple or multi-dimensional arrays

array_pad()	Inserts a specified number of items, with a specified value, to an array
array_pop()	Deletes the last element of an array
array_product()	Calculates the product of the values in an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reduce()	Returns an array as a string, using a user-defined function
array_replace()	Replaces the values of the first array with the values from following arrays
array_replace_recursive()	Replaces the values of the first array with the values from following arrays recursively
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_shift()	Removes the first element from an array, and returns the value of the removed element
array_slice()	Returns selected parts of an array
array_splice()	Removes and replaces specified elements of an array
array_sum()	Returns the sum of the values in an array
array_udiff()	Compare arrays, and returns the differences (compare values only, using a user-defined key comparison function)
array_uintersect()	Compare arrays, and returns the matches (compare values only, using a user-defined key comparison function)
array_unique()	Removes duplicate values from an array
array_unshift()	Adds one or more elements to the beginning of an array
array_values()	Returns all the values of an array
array_walk()	Applies a user function to every member of an array
arsort()	Sorts an associative array in descending order, according to the value
asort()	Sorts an associative array in ascending order, according to the value
compact()	Create array containing variables and their values

count()	Returns the number of elements in an array
current()	Returns the current element in an array
each()	Returns the current key and value pair from an array
end()	Sets the internal pointer of an array to its last element
extract()	Imports variables into the current symbol table from an array
in_array()	Checks if a specified value exists in an array
key()	Fetches a key from an array
list()	Assigns variables as if they were an array
natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
natsort()	Sorts an array using a "natural order" algorithm
next()	Advance the internal array pointer of an array
prev()	Rewinds the internal array pointer
range()	Creates an array containing a range of elements
reset()	Sets the internal pointer of an array to its first element
rsort()	Sorts an indexed array in descending order
shuffle()	Shuffles an array
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

- **Calendar Functions**

The calendar extension contains functions that simplify converting between different calendar formats. It is based on the Julian Day Count, which is a count of days starting from January 1st, 4713 B.C. To convert between calendar formats, first convert to Julian Day Count, then to the calendar of the user's choice.

Function	Description
cal_days_in_month()	Returns the number of days in a month for a specified year and calendar
cal_from_jd()	Converts a Julian Day Count into a date of a specified calendar

cal_info()	Returns information about a specified calendar
cal_to_jd()	Converts a date in a specified calendar to Julian Day Count
easter_date()	Returns the Unix timestamp for midnight on Easter of a specified year
easter_days()	Returns the number of days after March 21, that the Easter Day is in a specified year
gregoriantojd()	Converts a Gregorian date to a Julian Day Count
jddayofweek()	Returns the day of the week
jdmonthname()	Returns a month name
jdtogregorian()	Converts a Julian Day Count to a Gregorian date
jdtounix()	Converts Julian Day Count to Unix timestamp
jewishtojd()	Converts a Jewish date to a Julian Day Count
juliantojd()	Converts a Julian date to a Julian Day Count
unixtojd()	Converts Unix timestamp to Julian Day Count

- **Date Functions**

The date/time functions allow to get the date and time from the server on which PHP script runs. These functions depend on the locale settings of the server. Remember to take daylight saving time and leap years into consideration when working with these functions.

Function	Description
checkdate()	Validates a Gregorian date
date_add()	Adds days, months, years, hours, minutes, and seconds to a date
date_create_from_format()	Returns a new DateTime object formatted according to a specified format
date_create()	Returns a new DateTime object
date_date_set()	Sets a new date
date_default_timezone_get()	Returns the default timezone used by all date/time functions
date_default_timezone_set()	Sets the default timezone used by all date/time functions

date_diff()	Returns the difference between two dates
date_format()	Returns a date formatted according to a specified format
date_interval_format()	Formats the interval
date_isodate_set()	Sets the ISO date
date_modify()	Modifies the timestamp
date_parse()	Returns an associative array with detailed info about a specified date
date_sub()	Subtracts days, months, years, hours, minutes, and seconds from a date
date_sun_info()	Returns an array containing info about sunset/sunrise and twilight begin/end, for a specified day and location
date_sunrise()	Returns the sunrise time for a specified day and location
date_sunset()	Returns the sunset time for a specified day and location
date_time_set()	Sets the time
date()	Formats a local date and time
getdate()	Returns date/time information of a timestamp or the current local date/time
gettimeofday()	Returns the current time
gmdate()	Formats a GMT/UTC date and time
gmmktime()	Returns the Unix timestamp for a GMT date
gmstrftime()	Formats a GMT/UTC date and time according to locale settings
idate()	Formats a local time/date as integer
localtime()	Returns the local time
microtime()	Returns the current Unix timestamp with microseconds
mktime()	Returns the Unix timestamp for a date

strftime()	Formats a local time and/or date according to locale settings
time()	Returns the current time as a Unix timestamp
timezone_name_get()	Returns the name of the timezone
timezone_offset_get()	Returns the timezone offset from GMT
timezone_open()	Creates new DateTimeZone object
timezone_version_get()	Returns the version of the timezone db

- **Directory functions**

The directory function allows retrieving information about directories and their contents.

Function	Description
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes a directory handle
dir()	Returns an instance of the Directory class
getcwd()	Returns the current working directory
opendir()	Opens a directory handle
readdir()	Returns an entry from a directory handle
rewinddir()	Resets a directory handle
scandir()	Returns an array of files and directories of a specified directory

- **Error handling functions**

The error functions are used to deal with error handling and logging. The error functions allow us to define own error handling rules, and modify the way the errors can be logged. The logging functions allow us to send messages directly to other machines, emails, or system logs. The error reporting functions allow us to customize what level and kind of error feedback is given.

Function	Description
debug_backtrace()	Generates a backtrace
debug_print_backtrace()	Prints a backtrace
error_get_last()	Returns the last error that occurred
error_log()	Sends an error message to a log, to a file, or to a mail account
error_reporting()	Specifies which errors are reported
restore_error_handler()	Restores the previous error handler
restore_exception_handler()	Restores the previous exception handler
set_error_handler()	Sets a user-defined error handler function
set_exception_handler()	Sets a user-defined exception handler function
trigger_error()	Creates a user-level error message
user_error()	Alias of trigger_error()
debug_backtrace()	Generates a backtrace

- **File system Functions**

The file system functions allow the user to access and manipulate the file system.

Function	Description
basename()	Returns the filename component of a path
chgrp()	Changes the file group
chmod()	Changes the file mode
chown()	Changes the file owner
clearstatcache()	Clears the file status cache
copy()	Copies a file
dirname()	Returns the directory name component of a path
disk_free_space()	Returns the free space of a directory
disk_total_space()	Returns the total size of a directory

fclose()	Closes an open file
feof()	Tests for end-of-file on an open file
fflush()	Flushes buffered output to an open file
fgetc()	Returns a character from an open file
fgets()	Returns a line from an open file
fgetss()	Returns a line, with HTML and PHP tags removed, from an open file
file()	Reads a file into an array
file_exists()	Checks whether or not a file or directory exists
file_get_contents()	Reads a file into a string
file_put_contents()	Writes a string to a file
fileatime()	Returns the last access time of a file
filectime()	Returns the last change time of a file
filegroup()	Returns the group ID of a file
fileinode()	Returns the inode number of a file
filemtime()	Returns the last modification time of a file
fileowner()	Returns the user ID (owner) of a file
fileperms()	Returns the permissions of a file
filesize()	Returns the file size
filetype()	Returns the file type
flock()	Locks or releases a file
fnmatch()	Matches a filename or string against a specified pattern
fopen()	Opens a file or URL
fpassthru()	Reads from an open file, until EOF, and writes the result to the output buffer
fputcsv()	Formats a line as CSV and writes it to an open file
fread()	Reads from an open file
fscanf()	Parses input from an open file according to a specified format

fseek()	Seeks in an open file
fstat()	Returns information about an open file
ftell()	Returns the current position in an open file
ftruncate()	Truncates an open file to a specified length
fwrite()	Writes to an open file
glob()	Returns an array of filenames / directories matching a specified pattern
is_dir()	Checks whether a file is a directory
is_executable()	Checks whether a file is executable
is_file()	Checks whether a file is a regular file
is_link()	Checks whether a file is a link
is_readable()	Checks whether a file is readable
is_uploaded_file()	Checks whether a file was uploaded via HTTP POST
is_writable()	Checks whether a file is writeable
lchgrp()	Changes group ownership of symlink
lchown()	Changes user ownership of symlink
link()	Creates a hard link
linkinfo()	Returns information about a hard link
lstat()	Returns information about a file or symbolic link
mkdir()	Creates a directory
move_uploaded_file()	Moves an uploaded file to a new location
pathinfo()	Returns information about a file path
pclose()	Closes a pipe opened by popen()
popen()	Opens a pipe
readfile()	Reads a file and writes it to the output buffer
readlink()	Returns the target of a symbolic link
realpath()	Returns the absolute pathname
realpath_cache_get()	Returns realpath cache entries

realpath_cache_size()	Returns realpath cache size
rename()	Renames a file or directory
rewind()	Rewinds a file pointer
rmdir()	Removes an empty directory
set_file_buffer()	Sets the buffer size of an open file
stat()	Returns information about a file
symlink()	Creates a symbolic link
touch()	Sets access and modification time of a file
umask()	Changes file permissions for files
unlink()	Deletes a file

- **Math functions**

Function	Description
abs()	Returns the absolute (positive) value of a number
acos()	Returns the arc cosine of a number
acosh()	Returns the inverse hyperbolic cosine of a number
asin()	Returns the arc sine of a number
asinh()	Returns the inverse hyperbolic sine of a number
atan()	Returns the arc tangent of a number in radians
atan2()	Returns the arc tangent of two variables x and y
atanh()	Returns the inverse hyperbolic tangent of a number
bindec()	Converts a binary number to a decimal number
ceil()	Rounds a number up to the nearest integer
cos()	Returns the cosine of a number
cosh()	Returns the hyperbolic cosine of a number
decbin()	Converts a decimal number to a binary number
dechex()	Converts a decimal number to a hexadecimal number
decoct()	Converts a decimal number to an octal number
deg2rad()	Converts a degree value to a radian value

exp()	Calculates the exponent of e
expm1()	Returns exp(x) – 1
floor()	Rounds a number down to the nearest integer
fmod()	Returns the remainder of x/y
getrandmax()	Returns the largest possible value returned by rand()
hexdec()	Converts a hexadecimal number to a decimal number
hypot()	Calculates the hypotenuse of a right-angle triangle
max()	Returns the highest value in an array, or the highest value of several specified values
min()	Returns the lowest value in an array, or the lowest value of several specified values
octdec()	Converts an octal number to a decimal number
pi()	Returns the value of PI
pow()	Returns x raised to the power of y
rad2deg()	Converts a radian value to a degree value
rand()	Generates a random integer
round()	Rounds a floating-point number
sin()	Returns the sine of a number
sinh()	Returns the hyperbolic sine of a number
sqrt()	Returns the square root of a number
srand()	Seeds the random number generator
tan()	Returns the tangent of a number
tanh()	Returns the hyperbolic tangent of a number

- **String functions**

Function	Description
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts

convert_cyr_string()	Converts a string from one Cyrillic character-set to another
convert_uudecode()	Decodes a uuencoded string
convert_uuencode()	Encodes a string using the uuencode algorithm
count_chars()	Returns information about characters used in a string
crc32()	Calculates a 32-bit CRC for a string
crypt()	One-way string encryption (hashing)
echo()	Outputs one or more strings
explode()	Breaks a string into an array
fprintf()	Writes a formatted string to a specified output stream
hex2bin()	Converts a string of hexadecimal values to ASCII characters
html_entity_decode()	Converts HTML entities to characters
htmlentities()	Converts characters to HTML entities
implode()	Returns a string from the elements of an array
join()	Alias of implode()
lcfirst()	Converts the first character of a string to lowercase
levenshtein()	Returns the Levenshtein distance between two strings
localeconv()	Returns locale numeric and monetary formatting information
ltrim()	Removes whitespace or other characters from the left side of a string
number_format()	Formats a number with grouped thousands
ord()	Returns the ASCII value of the first character of a string
parse_str()	Parses a query string into variables
print()	Outputs one or more strings
printf()	Outputs a formatted string
rtrim()	Removes whitespace or other characters from the right side of a string
setlocale()	Sets locale information
sha1()	Calculates the SHA-1 hash of a string

sha1_file()	Calculates the SHA-1 hash of a file
similar_text()	Calculates the similarity between two strings
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_ireplace()	Replaces some characters in a string (case-insensitive)
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_shuffle()	Randomly shuffles all characters in a string
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strcspn()	Returns the number of characters found in a string before any part of some specified characters are found
stripos()	Returns the position of the first occurrence of a string inside another string (case-insensitive)
stristr()	Finds the first occurrence of a string inside another string (case-insensitive)
strlen()	Returns the length of a string
strncmp()	String comparison of the first n characters (case-sensitive)
strpbrk()	Searches a string for any of a set of characters
strpos()	Returns the position of the first occurrence of a string inside another string (case-sensitive)
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string

strrpos()	Finds the position of the last occurrence of a string inside another string (case-sensitive)
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strstr()	Finds the first occurrence of a string inside another string (case-sensitive)
strtok()	Splits a string into smaller strings
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
strtr()	Translates certain characters in a string
substr()	Returns a part of a string
substr_compare()	Compares two strings from a specified start position (binary safe and optionally case-sensitive)
substr_count()	Counts the number of times a substring occurs in a string
substr_replace()	Replaces a part of a string with another string
trim()	Removes whitespace or other characters from both sides of a string
vfprintf()	Writes a formatted string to a specified output stream
vprintf()	Outputs a formatted string
vsprintf()	Writes a formatted string to a variable
wordwrap()	Wraps a string to a given number of characters

- **Miscellaneous Functions**

Function	Description
connection_aborted()	Checks whether the client has disconnected
connection_status()	Returns the current connection status
connection_timeout()	Deprecated in PHP 4.0.5. Checks whether the script has timed out
constant()	Returns the value of a constant

define()	Defines a constant
defined()	Checks whether a constant exists
die()	Prints a message and exits the current script
eval()	Evaluates a string as PHP code
exit()	Prints a message and exits the current script
get_browser()	Returns the capabilities of the user's browser
halt_compiler()	Halts the compiler execution
pack()	Packs data into a binary string
php_check_syntax()	Deprecated in PHP 5.0.5
php_strip_whitespace()	Returns the source code of a file with PHP comments and whitespace removed
sleep()	Delays code execution for a number of seconds
sys_getloadavg()	Gets system load average
time_nanosleep()	Delays code execution for a number of seconds and nanoseconds
time_sleep_until()	Delays code execution until a specified time
uniqid()	Generates a unique ID
unpack()	Unpacks data from a binary string
usleep()	Delays code execution for a number of microseconds