PHP Array & String Function

Array Function	String Function
array_chunk	strlen
array_key_exists	str_word_count
array_keys	strrev
array_merge	strpos
array_push	str_replace
array_rand	strtoupper
array_slice	strtolower
array_values	ucfirst
sort	lcfirst
asort	ucwords
usort	addcslashes
uasort	addslashes
uksort	stripslashes
arsort	explode
ksort	implode
krsort	str_split
rsort	trim
count	ltrim
in_array	rtrim
array_search	strstr
	md5
	substr
	wordwrap
	nl2br

array_chunk

The **array_chunk()** function splits an array into chunks of new arrays.
syntex:- array_chunk(array, size, preserve_key)

```
<?php
$cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel");
print_r(array_chunk($cars,2));
?>
output:- Array (
[0] => Array ( [0] => Volvo [1] => BMW )
[1] => Array ( [0] => Toyota [1] => Honda )
[2] => Array ( [0] => Mercedes [1] => Opel )
)
```

array_key_exists

The **array_key_exists()** function checks an array for a specified key, and returns true if the key exists and false if the key does not exist.
syntex:- array_key_exists(key,array)

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5");
if (array_key_exists("Volvo",$a))
  {
  echo "Key exists!";
  }
else
  {
  echo "Key does not exist!";
  }
?>
output:- Key Exists!
```

array_keys

The **array_keys()** function returns an array containing the keys. **Syntex:-** array_keys(array,value,strict)

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
print_r(array_keys($a));
?>
output:-
Array
(
    [0] => Volvo
    [1] => BMW
    [2] => Toyota
)
```

array_merge

The **array_merge()** function merges one or more arrays into one array. **syntex:-** array_merge(array1,array2,array3...)

```
<?php
$a1=array("red","green");
$a2=array("blue","yellow");
print_r(array_merge($a1,$a2));
?>
output:-
Array(
[0] => red
[1] => green
[2] => blue
[3] => yellow
)
```

array_push

The **array_push()**__ function inserts one or more elements to the end of an array.
syntex:- array_push(array,value1,value2...)

```
<?php
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
?>
output:-
Array(
[0] => red
[1] => green
[2] => blue
[3] => yellow
)
```

array_rand

The **array_rand()** function returns a random key from an array, or it returns an array of random keys if you specify that the function should return more than one key.
syntex:- array_rand(array,number)

```
<?php
$a=array("red","green","blue","yellow","brown");
$random_keys=array_rand($a,3);
echo $a[$random_keys[0]]."<br>";
echo $a[$random_keys[1]]."<br>";
echo $a[$random_keys[2]];
?>
```

```
output:-
red
blue
yellow
```

array_slice

The **array_slice()** function returns selected parts of an array.
syntex:-array_slice(array,start,length,preserve)

```
<?php
$a=array("red","green","blue","yellow","brown");
print_r(array_slice($a,2));
?>
output:-
Array(
[0] => blue
[1] => yellow
[2] => brown
)
```

array_values

The **array_values()** function returns an array containing all the values of an array.
syntex:-array_values(array)

```
<?php
$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");
print_r(array_values($a));
?>
output:-
Array (
[0] => Peter
[1] => 41
[2] => USA
)
```

sort

The **sort()** function sorts an indexed array in ascending order.
syntex:-sort(array,sortingtype);

```
<?php
$cars=array("Volvo","BMW","Toyota");
sort($cars);
$clength=count($cars);
```

```
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
output:-
BMW
Toyota
Volvo
```

asort

The **asort()** function sorts an associative array in ascending order, according to the value.
syntex:-asort(array,sortingtype);

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
asort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

usort

The **usort()** function sorts an array using a user-defined comparison function.
syntex:-usort(array,myfunction);

```
<?php
function my_sort($a,$b)
{
if ($a==$b) return 0;
  return ($a<$b)?-1:1;
}
$a=array(4,2,8,6);
usort($a,"my_sort");

$arrlength=count($a);
for($x=0;$x<$arrlength;$x++)
  {
  echo $a[$x];
  echo "<br>";
  }
?>
output:-
2
4
6
8
```

uasort

The **uasort()** function sorts an array by values using a user-defined comparison function.
syntex:-uasort(array,myfunction);

```
<?php
function my_sort($a,$b)
{
if ($a==$b) return 0;
  return ($a<$b)?-1:1;
}
$arr=array("a"=>4,"b"=>2,"c"=>8,d=>"6");
uasort($arr,"my_sort");

foreach($arr as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=b, Value=2
Key=a, Value=4
Key=d, Value=6
Key=c, Value=8
```

uksort

The **uksort()** function sorts an array by keys using a user-defined comparison function.
syntex:-uksort(array,myfunction);

```
<?php
function my_sort($a,$b)
{
if ($a==$b) return 0;
  return ($a<$b)?-1:1;
}
$arr=array("a"=>4,"b"=>2,"c"=>8,d=>"6");
uksort($arr,"my_sort");
```

```
foreach($arr as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=a, Value=4
Key=b, Value=2
Key=c, Value=8
Key=d, Value=6
```

arsort

The **arsort()** function sorts an associative array in descending order, according to the value.
syntex:-arsort(array,sortingtype);

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
arsort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=Joe, Value=43
Key=Ben, Value=37
Key=Peter, Value=35
```

ksort

The **ksort()** function sorts an associative array in ascending order, according to the key.
syntex:-ksort(array,sortingtype);

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
ksort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=Ben, Value=37
```

```
Key=Joe, Value=43
Key=Peter, Value=35
```

krsort

The **krsort()** function sorts an associative array in descending order, according to the key.
syntex:-krsort(array,sortingtype);

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
krsort($age);

foreach($age as $x=>$x_value)
   {
   echo "Key=" . $x . ", Value=" . $x_value;
   echo "<br>";
   }
?>
output:-
Key=Peter, Value=35
Key=Joe, Value=43
Key=Ben, Value=37
```

rsort

The **rsort()** function sorts an indexed array in descending order.
syntex:-rsort(array,sortingtype);

```
<?php
$cars=array("Volvo","BMW","Toyota");
rsort($cars);

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>
output:-
Volvo
Toyota
BMW
```

count

The **count()** function returns the number of elements in an array.
syntex:-count(array,mode);

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
output:-
3
```

in_array

The **in_array()** function searches an array for a specific value.
syntex:-in_array(search,array,type)

```
<?php
$people = array("Peter", "Joe", "Glenn", "Cleveland");

if (in_array("Glenn", $people))
  {
  echo "Match found";
  }
else
  {
  echo "Match not found";
  }
?>
output:- Match found
```

array_search

The **array_search()** function search an array for a value and returns the key.
syntex:- array_search(value, array, strict)

```
<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_search("red",$a);
?>
output:- a
```

Ques. Difference between array_merge() and array_combine()?

- The **array_merge()** function is used to merge two or more arrays into a single array.

```
$array1 = array("subject1" => "Python","subject2" => "sql");
$array2 = array("subject3" => "c/c++","subject4" => "java");

$final = array_merge($array1, $array2);
print_r($final);

Output:-
Array
(
    [subject1] => Python
    [subject2] => sql
    [subject3] => c/c++
    [subject4] => java
)
```

- The **array_combine()** function is used to combine two arrays and create a new array by using one array for **keys** and another array for **values**.

```
$array1 = array("subject1" ,"subject2");
$array2 = array( "c/c++", "java");
$final = array_combine($array1, $array2);

print_r($final);

Array
(
    [subject1] => c/c++
    [subject2] => java
)
```

Ques. How to get second last element of array in php?

```
<?php
$array = array(5,6,70,10,36,2);
echo $array[count($array) -2];
?>

Output:- 36
```

Ques. How to get common values from two array in php?

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"black","g"=>"purple");
$a3=array("a"=>"red","b"=>"black","h"=>"yellow");
```

```
$result=array_intersect($a1,$a2,$a3);
print_r($result);
?>

Output:- Array ( [a] => red )
```

String Function

strlen

The PHP **strlen()** function returns the length of a string.

```
<?php
echo strlen("Hello world!");
?>
Output:-12
```

str_word_count

The PHP **str_word_count()** function counts the number of words in a string:

```
<?php
echo str_word_count("Hello world!");
?>
Output:- 2
```

strrev

The PHP **strrev()** function reverses a string:

```
<?php
echo strrev("Hello world!");
?>
Output:-dlrowolleH
```

strpos

The PHP **strpos()** function searches for a specific text within a string.

```
<?php
echo strpos("Hello world!", "world");
?>
Output:- 6
```

str_replace

The PHP **str_replace()** function replaces some characters with some other characters in a string.

```
<?php
echo str_replace("world", "Dolly", "Hello world!");
?>
Output:-Hello Dolly!
```

strtoupper

The **strtoupper()** function converts a string to uppercase.
syntex:-strtoupper(string)

```
<?php
echo strtoupper("Hello WORLD!");
?>
output:- HELLO WORLD!
```

strtolower

The **strtolower()** function converts a string to lowercase.
syntex:-strtolower(string)

```
<?php
echo strtolower("Hello WORLD.");
?>
output:- hello world.
```

ucfirst

The **ucfirst()** function converts the first character of a string to uppercase.
syntex:-ucfirst(string)

```
<?php
echo ucfirst("hello world!");
?>
output:- Hello world!
```

lcfirst

The **lcfirst()** function converts the first character of a string to lowercase.
syntex:-lcfirst(string)

```
<?php
echo lcfirst("Hello world!");
?>
output:- hello world!
```

ucwords

The **ucwords()** function converts the first character of each word in a string to uppercase.
syntex:-ucwords(string)

```
<?php
echo ucwords("hello world");
?>
output:- Hello World
```

addcslashes

The **addcslashes()** function returns a string with backslashes in front of the specified characters.
syntex:-addcslashes(string,characters)

```
<?php
$str = addcslashes("Hello World!","W");
echo($str);
?>
output:- Hello \World!
```

addslashes

The **addslashes()** function returns a string with backslashes in front of predefined characters.
syntex:-addslashes(string)

```
<?php
$str = addslashes('What does "yolo" mean?');
echo($str);
?>
output:- What does \"yolo\" mean?
```

stripslashes

The **stripslashes()** function removes backslashes
Syntax: - stripslashes(string)

```
<?php
echo stripslashes("Who\'s Peter Griffin?");
?>
output:- Who's Peter Griffin?
```

explode

The **explode()** function breaks a string into an array.
Syntax:-explode(separator,string,limit)

```
<?php
$str = "Hello world. It's a beautiful day.";
print_r (explode(" ",$str));
?>
output:- Array (
[0] => Hello
[1] => world.
[2] => It's
[3] => a
[4] => beautiful
[5] => day.
)
```

implode

The **implode()** function returns a string from the elements of an array.
syntex:-implode(separator,array)

```
<?php
$arr = array('Hello','World!','Beautiful','Day!');
echo implode(" ",$arr);
?>
output:-
Hello World! Beautiful Day!
```

str_split

The **str_split()** function splits a string into an array.
syntes:-str_split(string,length)

```
<?php
print_r(str_split("Hello"));
?>
output:- Array ( [0] => H [1] => e [2] => l [3] => l [4] => o )
```

trim

The **trim()** function removes whitespace and other predefined characters from both sides of a string.
syntex:-trim(string,charlist)

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo trim($str,"Hed!");
?>
output:-
Hello World!
llo Worl
```

ltrim

The **ltrim()** function removes whitespace or other predefined characters from the left side of a string.
syntex:-ltrim(string,charlist)

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo ltrim($str,"Hello");
?>
output:-
Hello World!
World!
```

rtrim

The **rtrim()** function removes whitespace or other predefined characters from the right side of a string.
syntex:-rtrim(string,charlist)

```
<?php
$str = "Hello World!";
echo $str . "<br>";
echo rtrim($str,"World!");
?>
output:-
Hello World!
Hello
```

strstr

The **strstr()** function searches for the first occurrence of a string inside another string.
Syntax:-strstr(string,search,before_search)

```
<?php
echo strstr("Hello world! mohit","world");
?>
output:- world! mohit
```

md5

The **md5()** function calculates the MD5 hash of a string.
Syntax:-md5(string,raw)

```
<?php
$str = "Hello";
echo md5($str);
?>
output:  8b1a9953c4611296a827abf8c47804d7
```

substr

The **substr()** function returns a part of a string.
Syntax:- substr(string,start,length)

```
<?php
echo substr("Hello world",4);
?>
output:- o world
```

wordwrap

The **wordwrap()** function wraps a string into new lines when it reaches a specific length.
Syntax:- wordwrap(string,width,break,cut)

```
<?php
$str = "An example of a long word is: Supercalifragulistic";
echo wordwrap($str,15,"<br>\n");
?>
output:-
An example of a
long word is:
Supercalifragulistic
```

nl2br

The **nl2br()** function inserts HTML line breaks (
or

) in front of each newline (\n) in a string. **syntex:-** nl2br(string,xhtml)

```
<?php
echo nl2br("One line.\nAnother line.");
?>
output:-
One line
Another line
```