PHP ARRAY FUNCTIONS

PHP indicates the earliest version of PHP that supports the function.

Function	Description
array()	Creates an array **Example** <?php $a=array("Dog","Cat","Horse"); print_r($a); ?>// Output:Array ([0] => Dog [1] => Cat [2] => Horse)
array_change_key_case( )	Returns an array with all keys in lowercase or uppercase **Example** <?php $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse"); print_r(array_change_key_case($a,CASE_UPPER)); ?> Output: Array ([A] => Cat [B] => Dog [C] => Horse)
array_chunk()	Splits an array into chunks of arrays **Example** <?php $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse","d"=>"Cow"); print_r(array_chunk($a,2)); ?> Output: Array ([0] => Array ([0] => Cat [1] => Dog) [1] => Array ([0] => Horse [1] => Cow))
array_combine()	Creates an array by using one array for keys and another for its values **Example** <?php

	$a1=array("a","b","c","d"); $a2=array("Cat","Dog","Horse","Cow"); print_r(array_combine($a1,$a2)); ?> **Output** Array ([a] => Cat [b] => Dog [c] => Horse [d] => Cow)
array_count_values()	Returns an array with the number of occurrences for each value **Example** <?php $a=array("Cat","Dog","Horse","Dog"); print_r(array_count_values($a)); ?> **Output**:Array ([Cat] => 1 [Dog] => 2 [Horse] => 1)
array_diff()	Compares array values, and returns the differences. **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(3=>"Horse",4=>"Dog",5=>"Fish"); print_r(array_diff($a1,$a2)); ?> **Output:**Array ([0] => Cat)
array_diff_assoc()	Compares array keys and values, and returns the differences **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(0=>"Rat",1=>"Horse",2=>"Dog"); $a3=array(0=>"Horse",1=>"Dog",2=>"Cat"); print_r(array_diff_assoc($a1,$a2,$a3)); ?>

	Output: Array ([0] => Cat [2] => Horse)
array_diff_key()	Compares array keys, and returns the differences **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(2=>"Bird",3=>"Rat",4=>"Fish"); $a3=array(5=>"Horse",6=>"Dog",7=>"Bird"); print_r(array_diff_key($a1,$a2,$a3)); ?> **Output**Array ([0] => Cat [1] => Dog)

array_diff_uassoc()	Compares array keys and values, with an additional user-made function check, and returns the differences
	Example
	<?php
	function myfunction($v1,$v2)
	{
	if ($v1===$v2){
	return 0;
	}
	if ($v1>$v2){
	return 1;
	}
	else {
	return -1;
	}
	}
	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse");
	$a2=array(3=>"Dog",1=>"Cat",5=>"Horse");
	print_r(array_diff_uassoc($a1,$a2,"myfunction"));
	?>

	OutputArray ([0] => Dog [2] => Horse)

array_diff_ukey()	Compares array keys, with an additional user-made function check, and returns the differences
	Example
	<?php
	function myfunction($v1,$v2){
	if ($v1===$v2){
	return 0; }
	if ($v1>$v2){
	return 1;
	}
	else {
	return -1;
	}
	}
	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse");
	$a2=array(3=>"Rat",1=>"Bird",5=>"Monkey");
	print_r(array_diff_ukey($a1,$a2,"myfunction"));
	?>
	OutputArray ([0] => Dog [2] => Horse)

array_fill()	Fills an array with values **Example** <?php $a=array_fill(2,3,"Dog"); print_r($a); ?> **Output**Array ([2] => Dog [3] => Dog [4] => Dog)
array_filter()	Filters elements of an array using a user-made function **Example** <?php

	function myfunction($v){
	if ($v==="Horse"){
	return true;
	}
	return false;
	}
	$a=array(0=>"Dog",1=>"Cat",2=>"Horse");
	print_r(array_filter($a,"myfunction"));
	?>
	OutputArray ([2] => Horse)

array_flip()	Exchanges all keys with their associated values in an array **Example** <?php $a=array(0=>"Dog",1=>"Cat",2=>"Horse"); print_r(array_flip($a)); ?> **Output**Array ([Dog] => 0 [Cat] => 1 [Horse] => 2)
array_intersect()	Compares array values, and returns the matches **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(3=>"Horse",4=>"Dog",5=>"Fish"); print_r(array_intersect($a1,$a2)); ?> **Output**Array ([1] => Dog [2] => Horse)
array_intersect_assoc()	Compares array keys and values, and returns the matches **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse");

	$a2=array(3=>"Horse",1=>"Dog",0=>"Cat"); print_r(array_intersect_assoc($a1,$a2)); ?>// **Output**Array ([0] => Cat [1] => Dog)
array_intersect_key()	Compares array keys, and returns the matches **Example** <?php $a1=array(0=>"Cat",1=>"Dog",2=>"Horse"); $a2=array(2=>"Bird",0=>"Cat",4=>"Fish"); print_r(array_intersect_key($a1,$a2)); ?> **Output**Array ([0] => Cat [2] => Horse)

array_intersect_uassoc()	Compares array keys and values, with an additional user-made function check, and returns the matches
	Example
	<?php
	function myfunction($v1,$v2){
	if ($v1===$v2){
	return 0;
	}
	if ($v1>$v2){
	return 1;
	}
	else {
	return -1;
	}
	}
	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse");
	$a2=array(3=>"Dog",1=>"Cat",5=>"Horse");
	print_r(array_intersect_uassoc($a1,$a2,"myfunction"));

	?> **Output**Array ([1] => Cat)

array_intersect_ukey()	Compares array keys, with an additional user-made function check, and returns the matches
	Example
	<?php
	function myfunction($v1,$v2){
	if ($v1===$v2){
	return 0;}
	if ($v1>$v2){
	return 1;
	}
	else{
	return -1;}
	}
	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse");
	$a2=array(3=>"Rat",1=>"Bird",5=>"Monkey");
	print_r(array_diff_ukey($a1,$a2,"myfunction"));
	?>
	outputArray ([0] => Dog [2] => Horse)

array_key_exists()	Checks if the specified key exists in the array
	Example
	<?php
	$a=array("a"=>"Dog","b"=>"Cat");
	if (array_key_exists("a",$a))
	{
	echo "Key exists!";
	}
	else
	{

	echo "Key does not exist!"; } ?> **Output**Key exists!
array_keys()	Returns all the keys of an array **Example** <?php $a=array("a"=>"Horse","b"=>"Cat","c"=>"Dog"); print_r(array_keys($a)); ?> **Output**Array ([0] => a [1] => b [2] => c)

array_map()	Sends each value of an array to a user-made function, which returns new values
	Example
	<?php
	function myfunction($v)
	{
	if ($v==="Dog"){
	return "Fido";
	}
	return $v;
	}
	$a=array("Horse","Dog","Cat");
	print_r(array_map("myfunction",$a));
	?>
	OutputArray ([0] => Horse [1] => Fido [2] => Cat)

array_merge()	Merges one or more arrays into one array **Example** <?php $a1=array("a"=>"Horse","b"=>"Dog"); $a2=array("c"=>"Cow","b"=>"Cat");

	print_r(array_merge($a1,$a2)); ?> OutputArray ([a] => Horse [b] => Cat [c] => Cow)
array_merge_recursive()	Merges one or more arrays into one array **Example** <?php $a1=array("a"=>"Horse","b"=>"Dog"); $a2=array("c"=>"Cow","b"=>"Cat"); print_r(array_merge_recursive($a1,$a2)); ?> **Output**Array ([a] => Horse [b] => Array ([0] => Dog [1] => Cat) [c] => Cow)

array_multisort()	Sorts multiple or multi-dimensional arrays
	
	Example
	
	<?php
	$a1=array("Dog","Cat");
	$a2=array("Fido","Missy");
	array_multisort($a1,$a2);
	print_r($a1);
	print_r($a2);
	?>
	
	OutputArray ([0] => Cat [1] => Dog) Array ([0] => Missy [1] => Fido)

array_pad()	Inserts a specified number of items, with a specified value, to an array **Example** <?php $a=array("Dog","Cat"); print_r(array_pad($a,5,0)); ?> **Output**Array ([0] => Dog [1] => Cat [2] => 0 [3] => 0 [4] => 0)
array_pop()	Deletes the last element of an array

	Example
	<?php
	$a=array("Dog","Cat","Horse");
	array_pop($a);
	print_r($a);
	?>
	OutputArray ([0] => Dog [1] => Cat)

array_product()	Calculates the product of the values in an array **Example** <?php $a=array(5,5); echo(array_product($a)); ?> **Output**25

array_push()	Inserts one or more elements to the end of an array
	Example
	<?php
	$a=array("Dog","Cat");
	array_push($a,"Horse","Bird");
	print_r($a);
	?>
	OutputArray ([0] => Dog [1] => Cat [2] => Horse [3] => Bird)

array_rand()	Returns one or more random keys from an array **Example** <?php $a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); print_r(array_rand($a,1)); ?> **Output**b
array_reduce()	Returns an array as a string, using a user-defined function

	Example
	<?php
	function myfunction($v1,$v2)
	{
	return $v1 . "-" . $v2;
	}
	$a=array("Dog","Cat","Horse");
	print_r(array_reduce($a,"myfunction"));
	?>
	Output-Dog-Cat-Horse

array_reverse()	Returns an array in the reverse order **Example** <?php $a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); print_r(array_reverse($a)); ?> OutputArray ([c] => Horse [b] => Cat [a] => Dog)
array_search()	Searches an array for a given value and returns the key **Example** <?php $a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); echo array_search("Dog",$a); ?> **Output**a
array_shift()	Removes the first element from an array, and returns the value of the removed element **Example** <?php $a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse"); echo array_shift($a); print_r ($a); ?> OutputDogArray ([b] => Cat [c] => Horse)

array_slice()	Returns selected parts of an array **Example** <?php $a=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird"); print_r(array_slice($a,1,2)); ?> OutputArray ([0] => Cat [1] => Horse)

array_splice()	Removes and replaces specified elements of an array
	Example
	<?php
	$a1=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird");
	$a2=array(0=>"Tiger",1=>"Lion");
	array_splice($a1,0,2,$a2);
	print_r($a1);
	?>
	Output
	Array ([0] => Tiger [1] => Lion [2] => Horse [3] => Bird)

array_sum()	Returns the sum of the values in an array **Example** <?php $a=array(0=>"5",1=>"15",2=>"25"); echo array_sum($a); ?> **Output** 45

array_udiff()	Compares array values in a user-made function and returns an array
	Example
	<?php
	function myfunction($v1,$v2)
	{
	if ($v1===$v2)
	{
	return 0;
	}
	return 1;
	}

	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse"); $a2=array(1=>"Cat",2=>"Dog",3=>"Fish"); print_r(array_udiff($a1,$a2,"myfunction")); ?> OutputArray ([c] => Horse)

array_udiff_assoc()	Compares array keys, and compares array values in a user-made function, and returns an array
	
	Example
	
	<?php
	function myfunction($v1,$v2)
	{
	if ($v1===$v2)
	  {
	  return 0;
	  }
	return 1;
	}
	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
	$a2=array("a"=>"Cat","b"=>"Horse","c"=>"Dog");
	print_r(array_udiff_assoc($a1,$a2,"myfunction"));
	?>
	
	OutputArray ([b] => Dog [c] => Horse)

array_udiff_uassoc()	Compares array keys and array values in user-made functions, and returns an array
	
	Example
	
	<?php
	function myfunction_key($v1,$v2)
	{
	if ($v1===$v2)
	  {
	  return 0;
	  }
	return 1;
	}
	
	function myfunction_value($v1,$v2)
	{
	if ($v1===$v2)
	  {
	  return 0;
	  }
	return 1;
	}

	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse"); $a2=array("a"=>"Cat","b"=>"Dog","c"=>"Fish"); print_r(array_udiff_uassoc($a1,$a2,"myfunction_key","myfunction_value")); ?> OutputArray ([c] => Horse)

array_uintersect()	Compares array values in a user-made function and returns an array
	
	Example
	
	<?php
	function myfunction($v1,$v2)
	{
	if ($v1===$v2)
	{
	return 0;
	}
	if ($v1 > $v2) return 1;
	{
	return -1;
	}
	return 1;
	}
	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
	$a2=array(1=>"Cat",2=>"Dog",3=>"Fish");
	print_r(array_uintersect($a1,$a2,"myfunction"));
	?>
	
	Output
	
	Array ([a] => Cat [b] => Dog)

array_uintersect_assoc()	Compares array keys, and compares array values in a user-made function, and returns an array
	
	Example
	
	<?php
	
	function myfunction($v1,$v2)
	
	{
	
	if ($v1===$v2)
	
	{
	
	return 0;
	
	}

	return 1;
	}
	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
	$a2=array("a"=>"Cat","b"=>"Horse","c"=>"Dog");
	print_r(array_uintersect_assoc($a1,$a2,"myfunction"));
	?>
	Output
	Array ([a] Cat)

array_uintersect_uassoc( )	Compares array keys and array values in user-made functions, and returns an array
	Example
	<?php
	function myfunc_key($v1,$v2)
	{
	if ($v1===$v2)
	{
	return 0;
	}
	return 1;
	}
	function myfunc_value($v1,$v2)
	{
	if ($v1===$v2)
	{
	return 0;
	}
	return 1;
	}
	$a1=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
	$a2=array("a"=>"Cat","b"=>"Dog","c"=>"Dog");
	print_r(array_uintersect_uassoc($a1,$a2,"myfunc_key","myfunc_value"));
	?>
	Output
	Array ([a] => Cat [b] => Dog)

array_unique()	Removes duplicate values from an array **Example** <?php $a=array("a"=>"Cat","b"=>"Dog","c"=>"Cat"); print_r(array_unique($a)); ?> Output

	Array ([a] => Cat [b] => Dog)

array_unshift()	Adds one or more elements to the beginning of an array
	
	Example
	
	<?php
	$a=array("a"=>"Cat","b"=>"Dog");
	array_unshift($a,"Horse");
	print_r($a);
	?>
	
	Output
	
	Array ([0] => Horse [a] => Cat [b] => Dog)

array_values()	Returns all the values of an array **Example** <?php $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse"); print_r(array_values($a)); ?> Output Array ([0] => Cat [1] => Dog [2] => Horse)

array_walk()	Applies a user function to every member of an array
	
	Example
	
	<?php
	function myfunction($value,$key)
	{
	echo "The key $key has the value $value
	";
	}
	$a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
	array_walk($a,"myfunction");
	?>
	
	Output
	
	The key a has the value Cat
	The key b has the value Dog
	The key c has the value Horse

array_walk_recursive()	Applies a user function recursively to every member of an array **Example** <?php function myfunction($value,$key)

	{
	echo "The key $key has the value $value
	";
	}
	$a1=array("a"=>"Cat","b"=>"Dog");
	$a2=array($a1,"1"=>"Bird","2"=>"Horse");
	array_walk_recursive($a2,"myfunction");
	?>
	
	Output
	
	The key a has the value Cat
	The key b has the value Dog
	The key 1 has the value Bird
	The key 2 has the value Horse

arsort()	Sorts an array in reverse order and maintain index association
	
	Example
	
	<?php
	$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");
	arsort($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([c] => Horse [a] => Dog [b] => Cat)

asort()	Sorts an array and maintain index association
	
	Example
	
	<?php
	$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");
	asort($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([b] => Cat [a] => Dog [c] => Horse)

compact()	Create array containing variables and their values **Example** <?php $firstname = "Peter"; $lastname = "Griffin"; $age = "38"; $result = compact("firstname", "lastname", "age"); print_r($result); ?>

	OutputArray ([firstname] => Peter [lastname] => Griffin [age] => 38)
count()	Counts elements in an array, or properties in an object **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); $result = count($people); echo $result; ?> Output4
current()	Returns the current element in an array **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . " "; ?> OutputPeter
each()	Returns the current key and value pair from an array **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); print_r (each($people)); ?> Output Array ([1] => Peter [value] => Peter [0] => 0 [key] => 0)
end()	Sets the internal pointer of an array to its last element **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . " "; echo end($people); ?> Output Peter Cleveland

extract()	Imports variables into the current symbol table from an array **Example** <?php $a = 'Original'; $my_array = array("a" => "Cat","b" => "Dog", "c" => "Horse"); extract($my_array); echo "\$a = $a; \$b = $b; \$c = $c"; ?> Output$a = Cat; $b = Dog; $c = Horse

in_array()	Checks if a specified value exists in an array
	
	Example
	
	<?php
	$people = array("Peter", "Joe", "Glenn", "Cleveland");
	if (in_array("Glenn",$people)){
	  echo "Match found";
	  }
	else{
	  echo "Match not found";
	  }
	?>
	
	OutputMatch found

key()	Fetches a key from an array **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo "The key from the current position is: " . key($people); ?> OutputThe key from the current position is: 0

krsort()	Sorts an array by key in reverse order
	
	Example
	
	<?php
	$my_array = array("a" => "Dog", "b" => "Cat","c" => "Horse");
	krsort($my_array);
	print_r($my_array);
	?>

	OutputArray ([c] => Horse [b] => Cat [a] => Dog)

ksort()	Sorts an array by key
	
	Example
	
	<?php
	$my_array = array("a" =>"Dog", "b" => "Cat", "c" =>"Horse");
	ksort($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([a] => Dog [b] => Cat [c] => Horse)

list()	Assigns variables as if they were an array **Example** <?php $my_array = array("Dog","Cat","Horse"); list($a, $b, $c) = $my_array; echo "I have several animals, a $a, a $b and a $c."; ?> OutputI have several animals, a Dog, a Cat and a Horse.

natcasesort()	Sorts an array using a case insensitive "natural order" algorithm
	
	Example
	
	<?php
	$temp_files = array("temp15.txt","Temp10.txt",
	"temp1.txt","Temp22.txt","temp2.txt");
	natsort($temp_files);
	echo "Natural order: ";
	print_r($temp_files);
	echo "
	";
	natcasesort($temp_files);
	echo "Natural order case insensitve: ";
	print_r($temp_files);
	?>
	
	OutputNatural order: Array ([1] => Temp10.txt [3] =>
	Temp22.txt [2] => temp1.txt [4] => temp2.txt [0] =>
	temp15.txt)
	Natural order case insensitve: Array ([2] => temp1.txt [4] =>
	temp2.txt [1] => Temp10.txt [0] => temp15.txt [3] =>
	Temp22.txt)

natsort()	Sorts an array using a "natural order" algorithm **Example**

	<?php
	$temp_files = array("temp15.txt","temp10.txt",
	"temp1.txt","temp22.txt","temp2.txt");
	
	sort($temp_files);
	echo "Standard sorting: ";
	print_r($temp_files);
	echo "
	";
	
	natsort($temp_files);
	echo "Natural order: ";
	print_r($temp_files);
	?>
	
	OutputStandard sorting: Array ([0] => temp1.txt [1] =>
	temp10.txt [2] => temp15.txt [3] => temp2.txt [4] =>
	temp22.txt)
	Natural order: Array ([0] => temp1.txt [3] => temp2.txt [1]
	=> temp10.txt [2] => temp15.txt [4] => temp22.txt)

next()	Advance the internal array pointer of an array **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . " "; echo next($people); ?> OutputPeter Joe
pos()	Alias of current() **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo pos($people) . " "; ?> OutputPeter
prev()	Rewinds the internal array pointer **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . " ";

	echo next($people) . " "; echo prev($people); ?> Output:Peter Joe Peter
range()	Creates an array containing a range of elements **Example** <?php $number = range(0,5); print_r ($number); ?> OutputArray ([0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5)
reset()	Sets the internal pointer of an array to its first element **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . " "; echo next($people) . " "; echo reset($people); ?> OutputPeter Joe Peter

rsort()	Sorts an array in reverse order
	
	Example
	
	<?php
	$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");
	rsort($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([0] => Horse [1] => Dog [2] => Cat)

shuffle()	Shuffles an array

	Example
	
	<?php
	$my_array = array("a" => "Dog", "b" => "Cat", "c" => "Horse");
	shuffle($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([0] => Horse [1] => Cat [2] => Dog)

sizeof()	Alias of count() **Example** <?php $people = array("Peter", "Joe", "Glenn", "Cleveland"); $result = sizeof($people); echo $result; ?> Output 4

sort()	Sorts an array
	
	Example
	
	<?php
	$my_array = array("a" => "Dog","b" => "Cat", "c" => "Horse");
	sort($my_array);
	print_r($my_array);
	?>
	
	OutputArray ([0] => Cat [1] => Dog [2] => Horse)

uasort()	Sorts an array with a user-defined function and maintain index association
	
	Example
	
	<?php
	function my_sort($a, $b)
	{
	if ($a == $b) return 0;
	return ($a > $b) ? -1 : 1;
	}
	$people = array("Swanson" => "Joe",
	"Griffin" => "Peter", "Quagmire" => "Glenn",
	"swanson" => "joe", "griffin" => "peter",
	"quagmire" => "glenn");
	uasort($people, "my_sort");

	print_r ($people); ?> OutputArray ([griffin] => peter [swanson] => joe [quagmire] => glenn [Griffin] => Peter [Swanson] => Joe [Quagmire] => Glenn)

uksort()	Sorts an array by keys using a user-defined function
	
	Example
	
	<?php
	function my_sort($a, $b){
	  if ($a == $b) return 0;
	  return ($a > $b) ? -1 : 1;
	  }
	$people = array("Swanson" => "Joe",
	"Griffin" => "Peter", "Quagmire" => "Glenn",
	"swanson" => "joe", "griffin" => "peter",
	"quagmire" => "glenn");
	uksort($people, "my_sort");
	print_r ($people);
	?>
	
	OutputArray ([swanson] => joe [quagmire] => glenn [griffin] => peter [Swanson] => Joe [Quagmire] => Glenn [Griffin] => Peter)

usort()	Sorts an array by values using a user-defined function
	
	Example
	
	<?php
	function my_sort($a, $b){
	  if ($a == $b) return 0;
	  return ($a > $b) ? -1 : 1;
	  }
	$arr = array("Peter", "glenn","Cleveland",
	"peter","cleveland", "Glenn");
	usort($arr, "my_sort");
	print_r ($arr);
	?>
	
	OutputArray ([0] => peter [1] => glenn [2] => cleveland [3] => Peter [4] => Glenn [5] => Cleveland)