The array_merge() function merges one or more arrays into one array.

Syntax

```
array_merge(array1, array2, array3, ...)
```

Parameter Values

Parameter	Description
array1	Required. Specifies an array
array2	Optional. Specifies an array
array3,...	Optional. Specifies an array

```
<!DOCTYPE html>
<html>
<body>

<?php
$a1=array("red","green");
$a2=array("blue","yellow");
print_r(array_merge($a1,$a2));
?>

</body>
</html>
```

Array ([0] => red [1] => green [2] => blue [3] => yellow)

The array_pop() function deletes the last element of an array.

Syntax

```
array_pop(array)
```

Parameter Values

Parameter	Description
array	Required. Specifies an array

```
<!DOCTYPE html>
<html>
<body>

<?php
$a=array("red","green","blue");
array_pop($a);
print_r($a);
?>

</body>
</html>
```

Array ([0] => red [1] => green)

The array_push() function inserts one or more elements to the end of an array.

Tip: You can add one value, or as many as you like.

Note: Even if your array has string keys, your added elements will always have numeric keys (See example below).

Syntax

```
array_push(array, value1, value2, ...)
```

Parameter Values

Parameter	Description
array	Required. Specifies an array
value1	Optional. Specifies the value to add (Required in PHP versions before 7.3)
value2	Optional. Specifies the value to add

```
<!DOCTYPE html>
<html>
<body>

<?php
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
?>

</body>
</html>
```

Array ([0] => red [1] => green [2] => blue [3] => yellow)

```
<!DOCTYPE html>
<html>
<body>

<?php
$a=array("a"=>"red","b"=>"green");
array_push($a,"blue","yellow");
print_r($a);
?>

</body>
</html>
```

Array ([a] => red [b] => green [0] => blue [1] => yellow)

The array_replace() function replaces the values of the first array with the values from following arrays.

Syntax

```
array_replace(array1, array2, array3, ...)
```

Parameter Values

Parameter	Description
array1	Required. Specifies an array
array2	Optional. Specifies an array which will replace the values of *array1*

```php
<!DOCTYPE html>
<html>
<body>

<?php
$a1=array("red","green");
$a2=array("blue","yellow");
print_r(array_replace($a1,$a2));
?>

</body>
</html>
```

Array ([0] => blue [1] => yellow)

The array_reverse() function returns an array in the reverse order.

Syntax

```
array_reverse(array, preserve)
```

Parameter Values

Parameter	Description
array	Required. Specifies an array
preserve	Optional. Specifies if the function should preserve the keys of the array or not. Possible values: • true • false

```
<html>
<body>

<?php
$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota");
print_r(array_reverse($a));
?>

</body>
</html>
```

Array ([c] => Toyota [b] => BMW [a] => Volvo)

The array_search() function search an array for a value and returns the key.

Syntax

```
array_search(value, array, strict)
```

Parameter Values

Parameter	Description
value	Required. Specifies the value to search for
array	Required. Specifies the array to search in
strict	Optional. If this parameter is set to TRUE, then this function will search for identical elements in the array. Possible values: • true • false - Default When set to true, the number 5 is not the same as the string 5 (See example 2)

```
<html>
<body>

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

</body>
</html>
```

a

The array_shift() function removes the first element from an array, and returns the value of the removed element.

Note: If the keys are numeric, all elements will get new keys, starting from 0 and increases by 1 (See example below).

Syntax

```
array_shift(array)
```

Parameter Values

Parameter	Description
array	Required. Specifies an array

```
<html>
<body>

<?php
$a=array("a"=>"red","b"=>"green","c"=>"blue");
echo array_shift($a)."<br>";
print_r ($a);
?>

</body>
</html>
```

```
red
Array ( [b] => green [c] => blue )
```

The array_slice() function returns selected parts of an array.

Note: If the array have string keys, the returned array will always preserve the keys (See example 4).

Syntax

```
array_slice(array, start, length, preserve)
```

Parameter Values

Parameter	Description
array	Required. Specifies an array
start	Required. Numeric value. Specifies where the function will start the slice. 0 = the first element. If this value is set to a negative number, the function will start slicing that far from the last element. -2 means start at the second last element of the array.
length	Optional. Numeric value. Specifies the length of the returned array. If this value is set to a negative number, the function will stop slicing that far from the last element. If this value is not set, the function will return all elements, starting from the position set by the start-parameter.
preserve	Optional. Specifies if the function should preserve or reset the keys. Possible values: • true - Preserve keys • false - Default. Reset keys

```
<html>
<body>

<?php
$a=array("red","green","blue","yellow","brown");
print_r(array_slice($a,2));
?>

</body>
</html>
```

Array ([0] => blue [1] => yellow [2] => brown)

The sort() function sorts an indexed array in ascending order.

Tip: Use the rsort() function to sort an indexed array in descending order.

Syntax

```
sort(array, sorttype)
```

Parameter Values

Parameter	Description
array	Required. Specifies the array to sort
sorttype	Optional. Specifies how to compare the array elements/items. Possible values: • 0 = SORT_REGULAR - Default. Compare items normally (don't change types) • 1 = SORT_NUMERIC - Compare items numerically • 2 = SORT_STRING - Compare items as strings

```
<html>
<body>

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

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>

</body>
</html>
```

BMW

Toyota

Volvo

The rsort() function sorts an indexed array in descending order.

Tip: Use the sort() function to sort an indexed array in ascending order.

Syntax

```
rsort(array, sorttype)
```

Parameter Values

Parameter	Description
array	Required. Specifies the array to sort
sorttype	Optional. Specifies how to compare the array elements/items. Possible values: • 0 = SORT_REGULAR - Default. Compare items normally (don't change types) • 1 = SORT_NUMERIC - Compare items numerically • 2 = SORT_STRING - Compare items as strings • 3 = SORT_LOCALE_STRING - Compare items as strings, based on current locale • 4 = SORT_NATURAL - Compare items as strings using natural ordering • 5 = SORT_FLAG_CASE -

```
<html>
<body>

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

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>

</body>
</html>
```

BMW

Toyota

Volvo

```
<!DOCTYPE html>
<html>
<body>

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

$clength=count($cars);
for($x=0;$x<$clength;$x++)
  {
  echo $cars[$x];
  echo "<br>";
  }
?>

</body>
</html>
```

Volvo

Toyota

BMW