PHP Array Functions

- array() creates an array, with keys and values. If you skip the keys when you specify an array, an integer key is generated, starting at 0 and increases by 1 for each value.
- **Syntax-** array(key => value)

Parameter	Description
Key	Optional. Specifies the key, of type numeric or string. If not set, an integer key is generated, starting at 0
Value	Required. Specifies the value

- **Example 1**

```
<?php
   $a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse");
    print_r($a);
 ?>
```

The output of the code above will be:

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

- **Example 2**

```
<?php
  $a=array("Dog","Cat","Horse");
  print_r($a);
 ?>
```

The output of the code above will be:

Array ([0] => Dog [1] => Cat [2] => Horse)

- Print_r() method displays an entire array.

- **Some Library Functions-**

1) array_change_key_case()- this function returns an array with all array KEYS in lower case or upper case.

- **Syntax-** array_change_key_case(array,case)

Parameter	Description
Array	Required. Specifies the array to use
Case	Optional. Possible values:

- ✓ CASE_LOWER - Default value. Returns the array key values in lower case.
- ✓ CASE_UPPER - Returns the array key values in upper case.

❖ **Note:** If two or more array keys will be the same after running this function, the latest array will override the other.

➢ **Example 1**

```
<?php
   $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse");
   print_r(array_change_key_case($a,CASE_UPPER));
?>
```

The output of the code above will be:

Array ([A] => Cat [B] => Dog [C] => Horse)

2) array_chunk()- this function splits an array into chunks of new arrays.

➢ **Syntax-** array_chunk(array,size,preserve_key)

Parameter	**Description**
Array	Required. Specifies the array to use
Size	Required. Specifies how many elements each new array will contain
preserve_key	Optional. Possible values: • true - Preserves the keys from the original array. • false - Default. Does not preserve the keys from the original array.

➢ **Example 1**

```
<?php
  $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse","d"=>"Cow");
  print_r(array_chunk($a,2));
?>
```

The output of the code above will be:

```
Array (
        [0] => Array ( [0] => Cat [1] => Dog )
        [1] => Array ( [0] => Horse [1] => Cow )
)
```

➢ **Example 2**

```
<?php
  $a=array("a"=>"Cat","b"=>"Dog","c"=>"Horse","d"=>"Cow");
  print_r(array_chunk($a,2,true));
?>
```

The output of the code above will be:

```
Array (
[0] => Array ( [a] => Cat [b] => Dog )
[1] => Array ( [c] => Horse [d] => Cow )
)
```

3) array_combine()- this function creates an array by combining two other arrays, where the first array is the keys, and the other array is the values.

- **Syntax-**array_combine(array1,array2)

Parameter Description

array1	Required. An array, specifying the keys
array2	Required. An array, specifying the values

- **Note:** Both parameters must have equal number of elements.

- **Example-**

```
<?php
  $a1=array("a","b","c","d");
  $a2=array("Cat","Dog","Horse","Cow");
  print_r(array_combine($a1,$a2));
?>
```

The output of the code above will be:

Array ([a] => Cat [b] => Dog [c] => Horse [d] => Cow)

4) array_count_values()- this function returns an array, where the keys are the original array's values, and the values is the number of occurrences.

- **Syntax-**array_count_values(array)

Parameter Description

array	Required. Specifying an array.

- **Example-**

```
<?php
  $a=array("Cat","Dog","Horse","Dog");
  print_r(array_count_values($a));
?>
```

The output of the code above will be:

Array ([Cat] => 1 [Dog] => 2 [Horse] => 1)

5) array_diff()- this function compares two or more arrays, and returns an array with the keys and values from the first array, only if the value is not present in any of the other arrays.

- **Syntax-**array_diff(array1,array2,array3...)

Parameter	**Description**
array1	Required. The first array is the array that the others will be compared with
array2	Required. An array to be compared with the first array
array3	Optional. An array to be compared with the first array

- **Note:** You can compare the first array with one array, or as many as you like.Only the value is used in the comparison.
- **Example-**

```
<?php
   $a1=array(0=>"Cat",1=>"Dog",2=>"Horse");
   $a2=array(3=>"Horse",4=>"Dog",5=>"Fish");
   print_r(array_diff($a1,$a2));
?>
```

The output of the code above will be:

Array ([0] => Cat)

6) array_diff_assoc()- this function compares two or more arrays, and returns an array with the keys and values from the first array, only if they are not present in any of the other arrays.

- **Syntax-**array_diff_assoc(array1,array2,array3...)

Parameter	**Description**
array1	Required. The first array is the array that the others will be compared with
array2	Required. An array to be compared with the first array
array3	Optional. An array to be compared with the first array

Note: You can compare the first array with one array, or as many as you like. Both the key and the value is used in the comparison.

- **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));
?>
```

The output of the code above will be:

Array ([0] => Cat [2] => Horse)

7) array_diff_key()- this function compares two or more arrays, and returns an array with the keys and values from the first array, only if the key is not present in any of the other arrays.

- **Syntax-** array_diff_key(array1,array2,array3...)

Parameter	**Description**
array1	Required. The first array is the array that the others will be compared with
array2	Required. An array to be compared with the first array
array3	Optional. An array to be compared with the first array

- **Note:** You can compare the first array with one array, or as many as you like. Only the key is used in the comparison.
- **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));
?>
```

The output of the code above will be:

Array ([0] => Cat [1] => Dog)

8) array_diff_uassoc()- this function compares two or more arrays, checking for differences, before comparing the keys in a user-defined function, then returns an array with the keys and values from the first array, if the function allows it.

- **Syntax-**array_diff_uassoc(array1,array2,array3...,function)

Parameter	**Description**
array1	Required. The first array is the array that the others will be compared with
array2	Required. An array to be compared with the first array
array3	Optional. An array to be compared with the first array
Function	Required. The name of the user-made function

Note: You can compare the first array with one array, or as many as you like. Both the key and the value is used in the automatic comparison, then, in the user-defined function, only the keys are being compared.

- **Example 1 -**

```
<?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"));
?>
```

The output of the code above will be:

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

- **Example 2-**

How to assign more than two arrays to the function

```
<?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");
$a3=array(6=>"Bird",0=>"Dog",5=>"Horse");
print_r(array_diff_uassoc($a1,$a2,$a3,"myfunction"));
?>
```

The output of the code above will be:

Array ([2] => Horse)

9) array_diff_ukey()- this function compares the keys in two or more arrays, checking for differences, before comparing the keys in a user-defined function, then returns an array with the keys and values from the first array, if the function allows it.

- **Syntax-** array_diff_ukey(array1,array2,array3...,function)

Parameter	**Description**
array1	Required. The first array is the array that the others will be compared with
array2	Required. An array to be compared with the first array
array3	Optional. An array to be compared with the first array
Function	Required. The name of the user-made function

- **Example 1 -**

```
<?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"));
?>
```

The output of the code above will be:

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

- **Example 2-**

How to assign more than two arrays to the function

```
<?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");
$a3=array(6=>"Dog",7=>"Donkey",0=>"Horse");
print_r(array_diff_ukey($a1,$a2,$a3,"myfunction"));
?>
```

The output of the code above will be:

Array ([2] => Horse)

10) **array_fill()-** this function returns an array filled with the values you describe.

- **Syntax-** array_fill(start,number,value)

Parameter	**Description**
Start	Required. A numeric value, specifies the starting index of the key
Number	Required. A numeric value, specifies the number of entries
Value	Required. Specifies the value to be inserted

- **Example-**

```
<?php
$a=array_fill(2,3,"Dog");
print_r($a);
?>
```

The output of the code above will be:

Array ([2] => Dog [3] => Dog [4] => Dog)

11) array_filter()- this function passes each value in the array to a user-made function, which returns either true or false, and returns an array only with the values that returned true.

- **Syntax-** array_filter(array,function)

Parameter	**Description**
array	Required. Specifies an array

function Required. Name of the 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"));
?>
```

The output of the code above will be:

Array ([2] => Horse)

12) array_flip() – this function returns an array with all the original keys as values, and all original values as keys.

- **Syntax-** array_flip(array)

Parameter Description

array Required. Specifies an array

- **Example-**

```
<?php
$a=array(0=>"Dog",1=>"Cat",2=>"Horse");
print_r(array_flip($a));
?>
```

The output of the code above will be:
Array ([Dog] => 0 [Cat] => 1 [Horse] => 2)

13) array_intersect() – this function compares **the values** of two (or more) arrays, and returns the matches.

- **Syntax:** array_intersect(array1,array2,array3...);
- **Exaple-**

```
<?php
$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");
$a2=array("e"=>"red","f"=>"green","g"=>"blue");
$result=array_intersect($a1,$a2);
print_r($result);
?>
```

- **Variations-**

array_intersect_assoc()	Compare arrays and returns the matches (compare keys and values)
array_intersect_key()	Compare arrays, and returns the matches (compare keys only)
array_intersect_uassoc()	Compare arrays, and returns the matches (compare keys and values, using a user-defined key comparison function)
array_intersect_ukey()	Compare arrays, and returns the matches (compare keys only, using a user-defined key comparison function)

14) **array_merge()-** this function merges one or more arrays into one array.

- **Syntax:** array_merge(array1,array2,array3...)
- **Example-**

```
<?php
$a1=array("a"=>"Horse","b"=>"Dog");
$a2=array("c"=>"Cow","b"=>"Cat");
print_r(array_merge($a1,$a2));
?>
```

❖ **Note:** If two or more array elements have the same key, the last one overrides the others. If you assign only one array to the array_merge() function, and the keys are integers, the function returns a new array with integer keys starting at 0 and increases by 1 for each value.

15) array_reverse()- this function returns an array in the reverse order.

- **Syntax:** array_reverse(array,preserve)

Parameter	**Description**
Array	Required. Specifies an array
Preserve	Optional. Possible values: • true • false Specifies if the function should preserve the array's keys or not.

- **Example-**

```
<?php
$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse");
print_r(array_reverse($a));
?>
```

The output of the code above will be:

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

16) array_search()- this function search an array for a value and returns the key.

- **Syntax:-** array_search(value,array,strict)
- **Example-**

```
<?php
  $a=array("a"=>"5","b"=>5,"c"=>"5");
  echo array_search(5,$a,true);
?>
```

17) array_sum()- this function returns the sum of all the values in the array.

- **Syntax:-** array_sum(array)
- **Example-**

```
<?php
   $a=array(0=>"5",1=>"15",2=>"25");
   echo array_sum($a);
?>
```

18) array_unique() –this function removes duplicate values from an array. If two or more array values are the same, the first appearance will be kept and the other will be removed.

- **Syntax:-** array_unique(array)
- **Example-**

```
<?php
  $a=array("a"=>"Cat","b"=>"Dog","c"=>"Cat");
  print_r(array_unique($a));
?>
```

19) count()- function returns the number of elements in an array.

- **Syntax:-** count(*array*,*mode*)
  Mode Optional. Specifies the mode. Possible values:

  0 - Default. Does not count all elements of multidimensional arrays

  1 - Counts the array recursively (counts all the elements of multidimensional arrays)

- **Example-**

```
<? php
$a2=array("abc","def");
echo count($a2,0);
?>
```

20) asort() – this function sorts an associative array in ascending order, according to the value.

- **Syntax:-** asort(array,sortingtype);

Parameter	**Description**
array	Required. Specifies the array to sort
sortingtype	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 -

➢ **Example-**

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

❖ **Note: 1) arsort()** Sorts an associative array in descending order, according to the value.
2) ksort() sort an associative array in ascending order, according to the key.