PHP array()

Definition and Usage

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 )
```

PHP array_change_key_case() Function

Definition and Usage

The array_change_key_case() 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 - Deafult value. Returns the array key values in lower case. • CASE_UPPER - Returns the array key values in upper case.

Tips and Notes

Note: If two or more array keys will be the same after running this function, the last array will override the others. (See example 2)

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 )
```

PHP array_chunk() Function

Definition and Usage

The array_chunk() 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 )
)
```

PHP array_combine() Function

Definition and Usage

The array_combine() 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

Tips and Notes

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 )
```

PHP array_count_values() Function

Definition and Usage

The array_count_values() 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 arrary.

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 )
```

PHP array_diff() Function

Definition and Usage

The array_diff() 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

Tips and Notes

Tip: You can compare the first array with one array, or as many as you like.

Note: 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 )
```

PHP array_fill() Function

Definition and Usage

The array_fill() 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 )
```

PHP array_filter() Function

Definition and Usage

The array_filter() 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 )
```

PHP array_flip() Function

Definition and Usage

The array_flip() 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 )
```

PHP array_keys() Function

Definition and Usage

The array_keys() function returns an array containing the keys.

Syntax

```
array_keys(array,value)
```

Parameter	Description
array	Required. Specifies an array
value	Optional. You can specify a value, then only the keys with this value are returned
strict	Optional. Used with the value parameter. Possible values: • true - Returns the keys with the specified value, depending on type: the number 5 is not the same as the string "5". • false - Default value. Not depending on type, the number 5 is the same as the string "5".

Example 1

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

The output of the code above will be:

```
Array ( [0] => a [1] => b [2] => c )
```

Example 2

Using the value parameter.

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

The output of the code above will be:

```
Array ( [0] => c)
```

Example 3

Using the strict parameter: false

```
<?php
$a=array(10,20,30,"10");
print_r(array_keys($a,"10",false));
?>
```

The output of the code above will be:

```
Array ( [0] => 0 [1] => 3 )
```

PHP array_map() Function

Definition and Usage

The array_map() function sends each value of an array to a user-made function, and returns an array with new values, given by the user-made function.

Syntax

```
array_map(function,array1,array2,array3...)
```

Parameter	Description
function	Required. The name of the user-made function, or null
array1	Required. Specifies an array
array2	Optional. Specifies an array
array3	Optional. Specifies an array

Tips and Notes

Tip: You can assign one array to the function, or as many as you like.

Example 1

```
<?php
function myfunction($v)
{
if ($v==="Dog")
        {
        return "Fido";
        }
return $v;
}
$a=array("Horse","Dog","Cat");
print_r(array_map("myfunction",$a));
?>
```

The output of the code above will be:

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

PHP array_merge() Function

Definition and Usage

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

Syntax

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

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

Tips and Notes

Tip: You can assign one array to the function, or as many as you like.

Note: If two or more array elements have the same key, the last one overrides the others.

Note: 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. (See example 2)

Example 1

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

The output of the code above will be:

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

PHP array_push() Function

Definition and Usage

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

Syntax

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

Parameter	Description
array	Required. Specifies an array
value1	Required. Specifies the value to add
value2	Optional. Specifies the value to add

Tips and Notes

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

Note: Even if your array has string keys, your added elements will allways have numeric keys. (See example 2)

Example 1

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

The output of the code above will be:

```
Array ( [0] => Dog [1] => Cat [2] => Horse [3] => Bird )
```

PHP array_pop() Function

Definition and Usage

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

Syntax

```
array_pop(array)
```

Parameter	Description
array	Required. Specifies an array

Example

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

The output of the code above will be:

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

PHP array_rand() Function

Definition and Usage

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.

Syntax

```
array_rand(array,number)
```

Parameter	Description
array	Required. Specifies an array
number	Optional. Default 1. Specifies how many random keys to return

Example 1

```
<?php
$a=array("a"=>"Dog","b"=>"Cat","c"=>"Horse");
print_r(array_rand($a,1));
?>
```

The output of the code above could be:

```
b
```

Example 2

An array with string keys:

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

The output of the code above could be:

```
Array ( [0] => c [1] => b )
```

PHP array_reverse() Function

Definition and Usage

The array_reverse() 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 )
```

PHP array_unique() Function

Definition and Usage

The array_unique() 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)
```

Parameter	Description
array	Required. Specifying an array

Tips and Notes

Note: The returned array will keep the **first** array item's key type.

Example

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

The output of the code above will be:

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

PHP array_slice() Function

Definition and Usage

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

Syntax

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

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. Possible values: • true - Preserve keys • false - Default - Reset keys

Tips and Notes

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

Example 1

```
<?php
$a=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird");
print_r(array_slice($a,1,2));
?>
```

The output of the code above will be:

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

Example 2

With a negative start parameter:

```
<?php
$a=array(0=>"Dog",1=>"Cat",2=>"Horse",3=>"Bird");
print_r(array_slice($a,-2,1));
?>
```

The output of the code above will be:

```
Array ( [0] => Horse )
```

PHP array_sum() Function

Definition and Usage

The array_sum() function returns the sum of all the values in the array.

Syntax

```
array_sum(array)
```

Parameter	Description
array	Required. Specifies an array

Example

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

The output of the code above will be:

```
45
```

PHP asort() Function

Definition and Usage

The asort() function sorts an array by the values. The values keep their original keys.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
asort(array,sorttype)
```

Parameter	Description
array	Required. Specifying an array
sorttype	Optional. Specifies how to sort the array values. Possible values: • SORT_REGULAR - Default. Treat values as they are (don't change types) • SORT_NUMERIC - Treat values numerically • SORT_STRING - Treat values as strings • SORT_LOCALE_STRING - Treat values as strings, based on local settings

Example

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

asort($my_array);
print_r($my_array);
?>
```

The output of the code above will be:

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

PHP arsort() Function

Definition and Usage

The arsort() function sorts an array by the values in reverse order. The values keep their original keys.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
arsort(array,sorttype)
```

Parameter	Description
array	Required. Specifying an array
sorttype	Optional. Specifies how to sort the array values. Possible values: • SORT_REGULAR - Default. Treat values as they are (don't change types) • SORT_NUMERIC - Treat values numerically • SORT_STRING - Treat values as strings • SORT_LOCALE_STRING - Treat values as strings, based on local settings

Example

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

arsort($my_array);
print_r($my_array);
?>
```

The output of the code above will be:

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

PHP sort() Function

Definition and Usage

The sort() function sorts an array by the values.

This function assigns new keys for the elements in the array. Existing keys will be removed.

This function returns TRUE on success, or FALSE on failure.

Syntax

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

Parameter	Description
array	Required. Specifies the array to sort
sorttype	Optional. Specifies how to sort the array values. Possible values: • SORT_REGULAR - Default. Treat values as they are (don't change types) • SORT_NUMERIC - Treat values numerically • SORT_STRING - Treat values as strings • SORT_LOCALE_STRING - Treat values as strings, based on local settings

Example

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

sort($my_array);
print_r($my_array);
?>
```

The output of the code above will be:

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

PHP rsort() Function

Definition and Usage

The rsort() function sorts an array by the values in reverse order.

This function assigns new keys for the elements in the array. Existing keys will be removed.

This function returns TRUE on success, or FALSE on failure.

Syntax

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

Parameter	Description
array	Required. Specifies the array to sort
sorttype	Optional. Specifies how to sort the array values. Possible values: • SORT_REGULAR - Default. Treat values as they are (don't change types) • SORT_NUMERIC - Treat values numerically • SORT_STRING - Treat values as strings • SORT_LOCALE_STRING - Treat values as strings, based on local settings

Example

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

rsort($my_array);
print_r($my_array);
?>
```

The output of the code above will be:

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

PHP count() Function

Definition and Usage

The count() function counts the elements of an array, or the properties of an object.

Syntax

```
count(array,mode)
```

Parameter	Description
array	Required. Specifies the array or object to count.
mode	Optional. Specifies the mode of the function. Possible values: • 0 - Default. Does not detect multidimensional arrays (arrays within arrays) • 1 - Detects multidimensional arrays **Note:** This parameter was added in PHP 4.2

Tips and Notes

Note: This function may return 0 if a variable isn't set, but it may also return 0 if a variable contains an empty array. The isset() function can be used to test if a variable is set.

Example

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

echo $result;
?>
```

The output of the code above will be:

```
4
```

PHP sizeof() Function

Definition and Usage

The sizeof() function counts the elements of an array, or the properties of an object.

This function is an alias of the count() function.

Syntax

```
sizeof(array1,array2)
```

Parameter	Description
array	Required. Specifies the array or object to count.
mode	Optional. Specifies the mode of the function. Possible values: • 0 - Default. Does not detect multidimensional arrays (arrays within arrays) • 1 - Detects multidimensional arrays **Note:** This parameter was added in PHP 4.2

Tips and Notes

Note: This function may return 0 if a variable isn't set, but it may also return 0 if a variable contains an empty array. The isset() function can be used to test if a variable is set.

Example

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

echo $result;
?>
```

The output of the code above will be:

```
4
```

PHP current() Function

Definition and Usage

The current() function returns the value of the current element in an array.

Syntax

```
current(array)
```

Parameter	Description
array	Required. Specifies the array to use

Tips and Notes

Note: This function returns FALSE on empty elements or elements with no value.

Tip: This function does not move the arrays internal pointer. To do this, use the next() and prev() functions.

Example 1

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

echo current($people) . "<br />";
?>
```

The output of the code above will be:

```
Peter
```

PHP end() Function

Definition and Usage

The end() function moves the internal pointer to, and outputs, the last element in the array.

This function returns the value of the last element in the array on success.

Syntax

```
end(array)
```

Parameter	Description
array	Required. Specifies the array to use

Example

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

echo current($people) . "<br />";
echo end($people);
?>
```

The output of the code above will be:

```
Peter
Cleveland
```

PHP shuffle() Function

Definition and Usage

The shuffle() function randomizes the order of the elements in the array.

This function assigns new keys for the elements in the array. Existing keys will be removed.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
shuffle(array)
```

Parameter	Description
array	Required. Specifies the array to use

Example

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

shuffle($my_array);
print_r($my_array);
?>
```

The output of the code above could be:

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

PHP range() Function

Definition and Usage

The range() function creates an array containing a range of elements.

This function returns an array of elements from low to high.

Syntax

```
range(low,high,step)
```

Parameter	Description
low	Required. Specifies the lowest value of the array
high	Required. Specifies the highest value of the array
step	Optional. Specifies the increment used in the range. Default is 1 **Note:** This parameter was added in PHP 5

Tips and Notes

Note: If the low parameter is higher than the high parameter, the range array will be from high to low.

Example 1

```
<?php
$number = range(0,5);
print_r ($number);
?>
```

The output of the code above will be:

```
Array
(
[0] => 0
[1] => 1
[2] => 2
[3] => 3
[4] => 4
[5] => 5
)
```

PHP reset() Function

Definition and Usage

The reset() function moves the internal pointer to the first element of the array.

This function returns the value of the first element in the array on success, or FALSE on failure.

Syntax

```
reset(array)
```

Parameter	Description
array	Required. Specifies the array to use

Example

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

echo current($people) . "<br />";
echo next($people) . "<br />";

echo reset($people);
?>
```

The output of the code above will be:

```
Peter
Joe
Peter
```