

Goto search
(current page)

/

Focus search box

[array_change_key_case »](#)

[« Sorting Arrays](#)

- [PHP Manual](#)
- [Function Reference](#)
- [Variable and Type Related Extensions](#)
- [Arrays](#)

Change language: English ▼

[Edit Report a Bug](#)

Array Functions ¶

See Also

See also [is_array\(\)](#), [explode\(\)](#), [implode\(\)](#), [split\(\)](#), [preg_split\(\)](#), and [unset\(\)](#).

Table of Contents ¶

- [array_change_key_case](#) — Changes the case of all keys in an array
- [array_chunk](#) — Split an array into chunks
- [array_column](#) — Return the values from a single column in the input array
- [array_combine](#) — Creates an array by using one array for keys and another for its values
- [array_count_values](#) — Counts all the values of an array
- [array_diff_assoc](#) — Computes the difference of arrays with additional index check
- [array_diff_key](#) — Computes the difference of arrays using keys for comparison
- [array_diff_uassoc](#) — Computes the difference of arrays with additional index check which is performed by a user supplied callback function
- [array_diff_ukey](#) — Computes the difference of arrays using a callback function on the keys for comparison
- [array_diff](#) — Computes the difference of arrays
- [array_fill_keys](#) — Fill an array with values, specifying keys
- [array_fill](#) — Fill an array with values
- [array_filter](#) — Filters elements of an array using a callback function
- [array_flip](#) — Exchanges all keys with their associated values in an array
- [array_intersect_assoc](#) — Computes the intersection of arrays with additional index check
- [array_intersect_key](#) — Computes the intersection of arrays using keys for comparison
- [array_intersect_uassoc](#) — Computes the intersection of arrays with additional index check, compares indexes by a callback function
- [array_intersect_ukey](#) — Computes the intersection of arrays using a callback function on the keys for comparison
- [array_intersect](#) — Computes the intersection of arrays
- [array_key_exists](#) — Checks if the given key or index exists in the array
- [array_keys](#) — Return all the keys or a subset of the keys of an array
- [array_map](#) — Applies the callback to the elements of the given arrays
- [array_merge_recursive](#) — Merge two or more arrays recursively

- [array_merge](#) — Merge one or more arrays
- [array_multisort](#) — Sort multiple or multi-dimensional arrays
- [array_pad](#) — Pad array to the specified length with a value
- [array_pop](#) — Pop the element off the end of array
- [array_product](#) — Calculate the product of values in an array
- [array_push](#) — Push one or more elements onto the end of array
- [array_rand](#) — Pick one or more random entries out of an array
- [array_reduce](#) — Iteratively reduce the array to a single value using a callback function
- [array_replace_recursive](#) — Replaces elements from passed arrays into the first array recursively
- [array_replace](#) — Replaces elements from passed arrays into the first array
- [array_reverse](#) — Return an array with elements in reverse order
- [array_search](#) — Searches the array for a given value and returns the corresponding key if successful
- [array_shift](#) — Shift an element off the beginning of array
- [array_slice](#) — Extract a slice of the array
- [array_splice](#) — Remove a portion of the array and replace it with something else
- [array_sum](#) — Calculate the sum of values in an array
- [array_udiff_assoc](#) — Computes the difference of arrays with additional index check, compares data by a callback function
- [array_udiff_uassoc](#) — Computes the difference of arrays with additional index check, compares data and indexes by a callback function
- [array_udiff](#) — Computes the difference of arrays by using a callback function for data comparison
- [array_uintersect_assoc](#) — Computes the intersection of arrays with additional index check, compares data by a callback function
- [array_uintersect_uassoc](#) — Computes the intersection of arrays with additional index check, compares data and indexes by separate callback functions
- [array_uintersect](#) — Computes the intersection of arrays, compares data by a callback function
- [array_unique](#) — Removes duplicate values from an array
- [array_unshift](#) — Prepend one or more elements to the beginning of an array
- [array_values](#) — Return all the values of an array
- [array_walk_recursive](#) — Apply a user function recursively to every member of an array
- [array_walk](#) — Apply a user supplied function to every member of an array
- [array](#) — Create an array
- [arsort](#) — Sort an array in reverse order and maintain index association
- [asort](#) — Sort an array and maintain index association
- [compact](#) — Create array containing variables and their values
- [count](#) — Count all elements in an array, or something in an object
- [current](#) — Return the current element in an array
- [each](#) — Return the current key and value pair from an array and advance the array cursor
- [end](#) — Set the internal pointer of an array to its last element
- [extract](#) — Import variables into the current symbol table from an array
- [in_array](#) — Checks if a value exists in an array
- [key_exists](#) — Alias of array_key_exists
- [key](#) — Fetch a key from an array
- [krsort](#) — Sort an array by key in reverse order
- [ksort](#) — Sort an array by key
- [list](#) — Assign variables as if they were an array
- [natcasesort](#) — Sort an array using a case insensitive "natural order" algorithm
- [natsort](#) — Sort an array using a "natural order" algorithm
- [next](#) — Advance the internal array pointer of an array
- [pos](#) — Alias of current
- [prev](#) — Rewind the internal array pointer
- [range](#) — Create an array containing a range of elements

- [reset](#) — Set the internal pointer of an array to its first element
- [rsort](#) — Sort an array in reverse order
- [shuffle](#) — Shuffle an array
- [sizeof](#) — Alias of count
- [sort](#) — Sort an array
- [uasort](#) — Sort an array with a user-defined comparison function and maintain index association
- [uksort](#) — Sort an array by keys using a user-defined comparison function
- [usort](#) — Sort an array by values using a user-defined comparison function

 [add a note](#)

User Contributed Notes 12 notes

[up](#)

[down](#)

12

[kolkabes at googlemail dot com ¶](#)

3 years ago

Short function for making a recursive array copy while cloning objects on the way.

```
<?php
function arrayCopy( array $array ) {
    $result = array();
    foreach( $array as $key => $val ) {
        if( is_array( $val ) ) {
            $result[$key] = arrayCopy( $val );
        } elseif ( is_object( $val ) ) {
            $result[$key] = clone $val;
        } else {
            $result[$key] = $val;
        }
    }
    return $result;
}
?>
```

[up](#)

[down](#)

12

[renatonascto at gmail dot com ¶](#)

7 years ago

Big arrays use a lot of memory possibly resulting in memory limit errors. You can reduce memory usage on your script by destroying them as soon as you're done with them. I was able to get over a few megabytes of memory by simply destroying some variables I didn't use anymore.

You can view the memory usage/gain by using the function `memory_get_usage()`. Hope this helps!

[up](#)

[down](#)

4

[dave at davidhbrown dot us ¶](#)

4 years ago

While PHP has well over three-score array functions, `array_rotate` is strangely missing as of PHP 5.3. Searching online offered several solutions, but the ones I found have defects such as inefficiently looping through the array or ignoring keys.