**List of All PHP Array Functions**

| **Function** | **Description** | **Syntax** |
| --- | --- | --- |
| **array\_change\_key\_case** | **Changes all keys in an array** | **array array\_change\_key\_case ( array $input [, int $case = CASE\_LOWER ] )** |
| **array\_chunk** | **Split an array into chunks** | **array array\_chunk ( array $input , int $size [, bool $preserve\_keys = false ] )** |
| **array\_combine** | **Creates an array by using one array for keys and another for its values** | **array array\_combine ( array $keys , array $values )** |
| **array\_count\_values** | **Counts all the values of an array** | **array array\_count\_values ( array $input )** |
| **array\_diff\_assoc** | **Computes the difference of arrays with additional index check** | **array array\_diff\_assoc ( array $array1 , array $array2 [, array $... ] )** |
| **array\_diff\_key** | **Computes the difference of arrays using keys for comparison** | **array array\_diff\_key ( array $array1 , array $array2 [, array $... ] )** |
| **array\_diff\_uassoc** | **Computes the difference of arrays with additional index check which is performed by a user supplied callback function** | **array array\_diff\_uassoc ( array $array1 , array $array2 [, array $... ], callback $key\_compare\_func )** |
| **array\_diff\_ukey** | **Computes the difference of arrays using a callback function on the keys for comparison** | **array array\_diff\_ukey ( array $array1 , array $array2 [, array $ ... ], callback $key\_compare\_func )** |
| **array\_diff** | **Computes the difference of arrays** | **array array\_diff ( array $array1 , array $array2 [, array $ ... ] )** |
| **array\_fill\_keys** | **Fill an array with values, specifying keys** | **array array\_fill\_keys ( array $keys , mixed $value )** |
| **array\_fill** | **Fill an array with values** | **array array\_fill ( int $start\_index , int $num , mixed $value )** |
| **array\_filter** | **Filters elements of an array using a callback function** | **array array\_filter ( array $input [, callback $callback ] )** |
| **array\_flip** | **Exchanges all keys with their associated values in an array** | **array array\_flip ( array $trans )** |
| **array\_intersect\_assoc** | **Computes the intersection of arrays with additional index check** | **array array\_intersect\_assoc ( array $array1 , array $array2 [, array $ ... ] )** |
| **array\_intersect\_key** | **Computes the intersection of arrays using keys for comparison** | **array array\_intersect\_key ( array $array1 , array $array2 [, array $ ... ] )** |
| **array\_intersect\_uassoc** | **Computes the intersection of arrays with additional index check, compares indexes by a callback function** | **array array\_intersect\_uassoc ( array $array1 , array $array2 [, array $ ... ], callback $key\_compare\_func )** |
| **array\_intersect\_ukey** | **Computes the intersection of arrays using a callback function on the keys for comparison** | **array array\_intersect\_ukey ( array $array1 , array $array2 [, array $... ], callback $key\_compare\_func )** |
| **array\_intersect** | **Computes the intersection of arrays** | **array array\_intersect ( array $array1 , array $array2 [, array $ ... ] )** |
| **array\_key\_exists** | **Checks if the given key or index exists in the array** | **bool array\_key\_exists ( mixed $key , array $search )** |
| **array\_keys** | **Return all the keys or a subset of the keys of an array** | **array array\_keys ( array $input [, mixed $search\_value [, bool $strict = false ]] )** |
| **array\_map** | **Applies the callback to the elements of the given arrays** | **array array\_map ( callback $callback , array $arr1 [, array $... ] )** |
| **array\_merge\_recursive** | **Merge two or more arrays recursively** | **array array\_merge\_recursive ( array $array1 [, array $... ] )** |
| **array\_merge** | **Merge one or more arrays** | **array array\_merge ( array $array1 [, array $... ] )** |
| **array\_multisort** | **Sort multiple or multi-dimensional arrays** | **bool array\_multisort ( array &$arr [, mixed $arg = SORT\_ASC [, mixed $arg = SORT\_REGULAR [, mixed $... ]]] )** |
| **array\_pad** | **Pad array to the specified length with a value** | **array array\_pad ( array $input , int $pad\_size , mixed $pad\_value )** |
| **array\_pop** | **Pop the element off the end of array** | **mixed array\_pop ( array &$array )** |
| **array\_product** | **Calculate the product of values in an array** | **number array\_product ( array $array )** |
| **array\_push** | **Push one or more elements onto the end of array** | **int array\_push ( array &$array , mixed $var [, mixed $... ] )** |
| **array\_rand** | **Pick one or more random entries out of an array** | **mixed array\_rand ( array $input [, int $num\_req = 1 ] )** |
| **array\_reduce** | **Iteratively reduce the array to a single value using a callback function** | **mixed array\_reduce ( array $input , callback $function [, mixed $initial = NULL ] )** |
| **array\_replace\_recursive** | **Replaces elements from passed arrays into the first array recursively** | **array array\_replace\_recursive ( array &$array , array &$array1 [, array &$... ] )** |
| **array\_replace** | **Replaces elements from passed arrays into the first array** | **array array\_replace ( array &$array , array &$array1 [, array &$... ] )** |
| **array\_reverse** | **Return an array with elements in reverse order** | **array array\_reverse ( array $array [, bool $preserve\_keys = false ] )** |
| **array\_search** | **Searches the array for a given value and returns the corresponding key if successful** | **mixed array\_search ( mixed $needle , array $haystack [, bool $strict = false ] )** |
| **array\_shift** | **Shift an element off the beginning of array** | **mixed array\_shift ( array &$array )** |
| **array\_slice** | **Extract a slice of the array** | **array array\_slice ( array $array , int $offset [, int $length [, bool $preserve\_keys = false ]] )** |
| **array\_splice** | **Remove a portion of the array and replace it with something else** | **array array\_splice ( array &$input , int $offset [, int $length = 0 [, mixed $replacement ]] )** |
| **array\_sum** | **Calculate the sum of values in an array** | **number array\_sum ( array $array )** |
| **array\_udiff\_assoc** | **Computes the difference of arrays with additional index check, compares data by a callback function** | **array array\_udiff\_assoc ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func )** |
| **array\_udiff\_uassoc** | **Computes the difference of arrays with additional index check, compares data and indexes by a callback function** | **array array\_udiff\_uassoc ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func , callback $key\_compare\_func )** |
| **array\_udiff** | **Computes the difference of arrays by using a callback function for data comparison** | **array array\_udiff ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func )** |
| **array\_uintersect\_assoc** | **Computes the intersection of arrays with additional index check, compares data by a callback function** | **array array\_uintersect\_assoc ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func )** |
| **array\_uintersect\_uassoc** | **Computes the intersection of arrays with additional index check, compares data and indexes by a callback functions** | **array array\_uintersect\_uassoc ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func , callback $key\_compare\_func )** |
| **array\_uintersect** | **Computes the intersection of arrays, compares data by a callback function** | **array array\_uintersect ( array $array1 , array $array2 [, array $ ... ], callback $data\_compare\_func )** |
| **array\_unique** | **Removes duplicate values from an array** | **array array\_unique ( array $array [, int $sort\_flags = SORT\_STRING ] )** |
| **array\_unshift** | **Prepend one or more elements to the beginning of an array** | **int array\_unshift ( array &$array , mixed $var [, mixed $... ] )** |
| **array\_values** | **Return all the values of an array** | **array array\_values ( array $input )** |
| **array\_walk\_recursive** | **Apply a user function recursively to every member of an array** | **bool array\_walk\_recursive ( array &$input , callback $funcname [, mixed $userdata ] )** |
| **array\_walk** | **Apply a user function to every member of an array** | **bool array\_walk ( array &$array , callback $funcname [, mixed $userdata ] )** |
| **array** | **Create an array** | **array array ([ mixed $... ] )** |
| **arsort** | **Sort an array in reverse order and maintain index association** | **bool arsort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **asort** | **Sort an array and maintain index association** | **bool asort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **compact** | **Create array containing variables and their values** | **array compact ( mixed $varname [, mixed $... ] )** |
| **count** | **Count all elements in an array, or properties in an object** | **int count ( mixed $var [, int $mode = COUNT\_NORMAL ] )** |
| **current** | **Return the current element in an array** | **mixed current ( array &$array )** |
| **each** | **Return the current key and value pair from an array and advance the array cursor** | **array each ( array &$array )** |
| **end** | **Set the internal pointer of an array to its last element** | **mixed end ( array &$array )** |
| **extract** | **Import variables into the current symbol table from an array** | **int extract ( array &$var\_array [, int $extract\_type = EXTR\_OVERWRITE [, string $prefix ]] )** |
| **in\_array** | **Checks if a value exists in an array** | **bool in\_array ( mixed $needle , array $haystack [, bool $strict = FALSE ] )** |
| **key** | **Fetch a key from an array** | **mixed key ( array &$array )** |
| **krsort** | **Sort an array by key in reverse order** | **bool krsort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **ksort** | **Sort an array by key** | **bool ksort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **list** | **Assign variables as if they were an array** | **array list ( mixed $varname [, mixed $... ] )** |
| **natcasesort** | **Sort an array using a case insensitive "natural order" algorithm** | **bool natcasesort ( array &$array )** |
| **natsort** | **Sort an array using a "natural order" algorithm** | **bool natsort ( array &$array )** |
| **next** | **Advance the internal array pointer of an array** | **mixed next ( array &$array )** |
| **pos** | **Alias of current()** |  |
| **prev** | **Rewind the internal array pointer** | **mixed prev ( array &$array )** |
| **range** | **Create an array containing a range of elements** | **array range ( mixed $start , mixed $limit [, number $step = 1 ] )** |
| **reset** | **Set the internal pointer of an array to its first element** | **mixed reset ( array &$array )** |
| **rsort** | **Sort an array in reverse order** | **bool rsort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **shuffle** | **Shuffle an array** | **bool shuffle ( array &$array )** |
| **sizeof** | **Alias of count()** |  |
| **sort** | **Sort an array** | **bool sort ( array &$array [, int $sort\_flags = SORT\_REGULAR ] )** |
| **uasort** | **Sort an array with a user-defined comparison function and maintain index association** | **bool uasort ( array &$array , callback $cmp\_function )** |
| **uksort** | **Sort an array by keys using a user-defined comparison function** | **bool uksort ( array &$array , callback $cmp\_function )** |
| **usort** | **Sort an array by values using a user-defined comparison function** | **bool usort ( array &$array , callback $cmp\_function )** |