|  |  |
| --- | --- |
| **Function** | **Description** |
| array() | Creates an array |
| array\_change\_key\_case()  array\_change\_key\_case(array,case);  Case:   * CASE\_LOWER - Default value. Changes the keys to lowercase * CASE\_UPPER - Changes the keys to uppercase | Changes all keys in an array to lowercase or uppercase  <?php $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43"); print\_r(array\_change\_key\_case($age,CASE\_UPPER)); ?>  **Output**  Array ( [PETER] => 35 [BEN] => 37 [JOE] => 43 ) |
| array\_chunk()  array\_chunk(array,size); | Splits an array into chunks of arrays  <?php $cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel"); print\_r(array\_chunk($cars,2)); ?>  **Output**  Array ( [0] => Array ( [0] => Volvo [1] => BMW ) [1] => Array ( [0] => Toyota [1] => Honda ) [2] => Array ( [0] => Mercedes [1] => Opel ) ) |
| array\_combine()  array\_combine(keys array,values array); | Creates an array by using the elements from one "keys" array and one "values" array  <?php $fname=array("Peter","Ben","Joe"); $age=array("35","37","43");  $c=array\_combine($fname,$age); print\_r($c); ?>  **Output**  Array ( [Peter] => 35 [Ben] => 37 [Joe] => 43 ) |
| array\_fill()  array\_fill(index,number,value); | Fills an array with values  <?php $a1=array\_fill(3,4,"blue"); print\_r($a1); ?>  **Output**  Array ( [3] => blue [4] => blue [5] => blue [6] => blue )  Array ( [0] => red ) |
| array\_flip() | Flips/Exchanges all keys with their associated values in an array  <?php $a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow"); $result=array\_flip($a1); print\_r($result); ?>  **output**  Array ( [red] => a [green] => b [blue] => c [yellow] => d ) |
| array\_intersect() | Compare arrays, and returns the matches (compare values only)  This function compares the values of two or more arrays, and return an array that contains the entries from array1 that are present in array2, array3, etc.  <?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); ?>  **Output**  Array ( [a] => red [b] => green [c] => blue ) |
| array\_key\_exists() | Checks if the specified key exists in the array  The array\_key\_exists() function checks an array for a specified key, and returns true if the key exists and false if the key does not exist.  <?php $a=array("Volvo"=>"XC90","BMW"=>"X5"); if (array\_key\_exists("Volvo",$a))   {   echo "Key exists!";   } else   {   echo "Key does not exist!";   } ?>  **Output**  Key exists! |
| array\_keys() | Returns all the keys of an array  <?php $a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander"); print\_r(array\_keys($a)); ?>  **Output**  Array ( [0] => Volvo [1] => BMW [2] => Toyota ) |
| array\_merge() | Merges one or more arrays into one array  <?php $a1=array("red","green"); $a2=array("blue","yellow"); print\_r(array\_merge($a1,$a2)); ?>  **Ouput**  Array ( [0] => red [1] => green [2] => blue [3] => yellow ) |
| array\_multisort()  array\_multisort(array1,sorting order,sorting type,array2,array3...)  *sorting order* Optional. Specifies the sorting order. Possible values:   * SORT\_ASC - Default. Sort in ascending order (A-Z) * SORT\_DESC - Sort in descending order (Z-A)   *sorting type* Optional. Specifies the type to use, when comparing elements. Possible values:   * SORT\_REGULAR - Default. Compare elements normally (Standard ASCII) * SORT\_NUMERIC - Compare elements as numeric values * SORT\_STRING - Compare elements as string values | Sorts multiple or multi-dimensional arrays  <?php $a1=array("Dog","Cat"); $a2=array("Fido","Missy"); array\_multisort($a1,$a2); print\_r($a1); print\_r($a2); ?>  **Output**  Array ( [0] => Cat [1] => Dog ) Array ( [0] => Missy [1] => Fido ) |
| array\_pop() | Deletes the last element of an array  <?php $a=array("red","green","blue"); array\_pop($a); print\_r($a); ?>  **Output**  Array ( [0] => red [1] => green ) |
| array\_product() | Calculates the product of the values in an array  <?php $a=array(5,5); echo(array\_product($a)); ?>  **Output**  **25** |
| array\_push() | Inserts one or more elements to the end of an array and Returns the new number of elements in the array  <?php $a=array("red","green"); array\_push($a,"blue","yellow"); print\_r($a); ?>  **Output**  Array ( [0] => red [1] => green [2] => blue [3] => yellow ) |
| array\_replace() | Replaces the values of the first array with the values from following arrays  <?php $a1=array("red","green"); $a2=array("blue","yellow"); print\_r(array\_replace($a1,$a2)); ?>  **Output**  Array ( [0] => blue [1] => yellow ) |
| [array\_reverse()](https://www.w3schools.com/php/func_array_reverse.asp) | Returns an array in the reverse order  <?php $a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota"); print\_r(array\_reverse($a)); ?>  **Output**  Array ( [c] => Toyota [b] => BMW [a] => Volvo ) |
| array\_search()  array\_search(value,array,strict)  ***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** | Searches an array for a given value and returns the key and FALSE otherwise. If the value is found in the array more than once, the first matching key is returned.  <?php $a=array("a"=>"red","b"=>"green","c"=>"blue"); echo array\_search("red",$a); ?>  **Output**  A |
| array\_shift() | Removes the first element from an array, and returns the value of the removed element  <?php $a=array("a"=>"red","b"=>"green","c"=>"blue"); echo array\_shift($a); print\_r ($a); ?>  **Output**  red Array ( [b] => green [c] => blue ) |
| array\_slice()  array\_slice(array,start,length) | Returns selected parts of an array  <?php $a=array("red","green","blue","yellow","brown"); print\_r(array\_slice($a,2)); ?>  **Output**  Array ( [0] => blue [1] => yellow [2] => brown ) |
| array\_splice()  array\_splice(array,start,length,array) | Removes and replaces specified elements of an array  <?php $a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow"); $a2=array("a"=>"purple","b"=>"orange"); array\_splice($a1,0,2,$a2); print\_r($a1); ?>  **Output**  Array ( [0] => purple [1] => orange [c] => blue [d] => yellow ) |
| array\_sum() | Returns the sum of the values in an array  <?php $a=array(5,15,25); echo array\_sum($a); ?>  **Output**  **45** |
| array\_unique() | Removes duplicate values from an array  <?php $a=array("a"=>"red","b"=>"green","c"=>"red"); print\_r(array\_unique($a)); ?>  **Output**  Array ( [a] => red [b] => green ) |
| array\_unshift() | Adds one or more elements to the beginning of an array  <?php $a=array("a"=>"red","b"=>"green"); array\_unshift($a,"blue"); print\_r($a); ?>  **Output**  Array ( [0] => blue [a] => red [b] => green ) |
| array\_values() | Returns all the values of an array |
| arsort() | Sorts an associative array in descending order, according to the value  <?php $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43"); arsort($age); ?>  **Output**  Key=Joe, Value=43 Key=Ben, Value=37 Key=Peter, Value=35 |
| asort() | Sorts an associative array in ascending order, according to the value |
| count() | Returns the number of elements in an array |
| current() or pos() | Returns the current element in an array |
| end() | Sets the internal pointer of an array to its last element |
| in\_array()  in\_array(search,array) | Checks if a specified value exists in an array and returns TRUE or FALSE accordingly |
| key() | Fetches a key from an array  <?php $people=array("Peter","Joe","Glenn","Cleveland"); echo "The key from the current position is: " . key($people); ?>  **Output**  The key from the current position is: 0 |
| krsort() | Sorts an associative array in descending order, according to the key |
| ksort() | Sorts an associative array in ascending order, according to the key |
| list() | Assigns variables as if they were an array  <?php $my\_array = array("Dog","Cat","Horse");  list($a, $b, $c) = $my\_array; echo "I have several animals, a $a, a $b and a $c."; ?>  **Output**  I have several animals, a Dog, a Cat and a Horse. |
| next() | Advance the internal array pointer of an array |
| prev() | Rewinds the internal array pointer |
| range()  range(low,high,step) | Creates an array containing a range of elements  <?php $number = range(0,5); print\_r ($number); ?>  **Output**  Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 ) |
| reset() | Sets the internal pointer of an array to its first element |
| rsort() | Sorts an indexed array in descending order |
| sizeof() | Alias of count() |
| [sort()](https://www.w3schools.com/php/func_array_sort.asp) | Sorts an indexed array in ascending order |
| uasort() | Sorts an array by values using a user-defined comparison function |
| uksort() | Sorts an array by keys using a user-defined comparison function |
| usort() | Sorts an array using a user-defined comparison function  <?php  function mysort($a,$b)  {  return $a-$b;  }  $a=array(4,7,2,9,10);  usort($a,"mysort");  print\_r($a);  ?>  //Output  Array ( [0] => 2 [1] => 4 [2] => 7 [3] => 9 [4] => 10 ) |