

## PHP ARRAY FUNCTIONS

PHP indicates the earliest version of PHP that supports the function.

Function	Description
array()	<p>Creates an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("Dog","Cat","Horse"); print_r(\$a); ?&gt;</pre> <p>// Output:Array ( [0] =&gt; Dog [1] =&gt; Cat [2] =&gt; Horse )</p>
array_change_key_case( )	<p>Returns an array with all keys in lowercase or uppercase</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); print_r(array_change_key_case(\$a,CASE_UPPER)); ?&gt;</pre> <p>Output:</p> <p>Array ( [A] =&gt; Cat [B] =&gt; Dog [C] =&gt; Horse )</p>
array_chunk()	<p>Splits an array into chunks of arrays</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse","d"=&gt;"Cow"); print_r(array_chunk(\$a,2)); ?&gt;</pre> <p>Output:</p> <p>Array ( [0] =&gt; Array ( [0] =&gt; Cat [1] =&gt; Dog ) [1] =&gt; Array ( [0] =&gt; Horse [1] =&gt; Cow ) )</p>
array_combine()	<p>Creates an array by using one array for keys and another for its values</p> <p><b>Example</b></p> <pre>&lt;?php</pre>

	<pre>\$a1=array("a","b","c","d"); \$a2=array("Cat","Dog","Horse","Cow"); print_r(array_combine(\$a1,\$a2)); ?&gt;</pre> <p><b>Output</b></p> <p>Array ( [a] =&gt; Cat [b] =&gt; Dog [c] =&gt; Horse [d] =&gt; Cow )</p>
array_count_values()	<p>Returns an array with the number of occurrences for each value</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("Cat","Dog","Horse","Dog"); print_r(array_count_values(\$a)); ?&gt;</pre> <p><b>Output:</b>Array ( [Cat] =&gt; 1 [Dog] =&gt; 2 [Horse] =&gt; 1 )</p>
array_diff()	<p>Compares array values, and returns the differences.</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse"); \$a2=array(3=&gt;"Horse",4=&gt;"Dog",5=&gt;"Fish"); print_r(array_diff(\$a1,\$a2)); ?&gt;</pre> <p><b>Output:</b>Array ( [0] =&gt; Cat )</p>
array_diff_assoc()	<p>Compares array keys and values, and returns the differences</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse"); \$a2=array(0=&gt;"Rat",1=&gt;"Horse",2=&gt;"Dog"); \$a3=array(0=&gt;"Horse",1=&gt;"Dog",2=&gt;"Cat"); print_r(array_diff_assoc(\$a1,\$a2,\$a3)); ?&gt;</pre>

	<p><b>Output:</b>Array ( [0] =&gt; Cat [2] =&gt; Horse )</p>
<p>array_diff_key()</p>	<p>Compares array keys, and returns the differences</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse"); \$a2=array(2=&gt;"Bird",3=&gt;"Rat",4=&gt;"Fish"); \$a3=array(5=&gt;"Horse",6=&gt;"Dog",7=&gt;"Bird"); print_r(array_diff_key(\$a1,\$a2,\$a3)); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Cat [1] =&gt; Dog )</p>
<p>array_diff_uassoc()</p>	<p>Compares array keys and values, with an additional user-made function check, and returns the differences</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { if (\$v1=== \$v2){ return 0; } if (\$v1&gt;\$v2){ return 1; } else { return -1; } } \$a1=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); \$a2=array(3=&gt;"Dog",1=&gt;"Cat",5=&gt;"Horse"); print_r(array_diff_uassoc(\$a1,\$a2,"myfunction")); ?&gt;</pre>

	<p><b>Output</b>Array ( [0] =&gt; Dog [2] =&gt; Horse )</p>
array_diff_ukey()	<p>Compares array keys, with an additional user-made function check, and returns the differences</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2){ if (\$v1=== \$v2){ return 0; } if (\$v1&gt;\$v2){ return 1; } else { return -1; } } \$a1=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); \$a2=array(3=&gt;"Rat",1=&gt;"Bird",5=&gt;"Monkey"); print_r(array_diff_ukey(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Dog [2] =&gt; Horse )</p>
array_fill()	<p>Fills an array with values</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array_fill(2,3,"Dog"); print_r(\$a); ?&gt;</pre> <p><b>Output</b>Array ( [2] =&gt; Dog [3] =&gt; Dog [4] =&gt; Dog )</p>
array_filter()	<p>Filters elements of an array using a user-made function</p> <p><b>Example</b></p> <pre>&lt;?php</pre>

	<pre>function myfunction(\$v){ if (\$v==="Horse"){     return true; } return false; } \$a=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); print_r(array_filter(\$a,"myfunction")); ?&gt;</pre> <p><b>Output</b>Array ( [2] =&gt; Horse )</p>
array_flip()	<p>Exchanges all keys with their associated values in an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); print_r(array_flip(\$a)); ?&gt;</pre> <p><b>Output</b>Array ( [Dog] =&gt; 0 [Cat] =&gt; 1 [Horse] =&gt; 2 )</p>
array_intersect()	<p>Compares array values, and returns the matches</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse"); \$a2=array(3=&gt;"Horse",4=&gt;"Dog",5=&gt;"Fish"); print_r(array_intersect(\$a1,\$a2)); ?&gt;</pre> <p><b>Output</b>Array ( [1] =&gt; Dog [2] =&gt; Horse )</p>
array_intersect_assoc()	<p>Compares array keys and values, and returns the matches</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse");</pre>

	<pre>\$a2=array(3=&gt;"Horse",1=&gt;"Dog",0=&gt;"Cat"); print_r(array_intersect_assoc(\$a1,\$a2)); ?&gt;// <b>Output</b>Array ( [0] =&gt; Cat [1] =&gt; Dog )</pre>
<p>array_intersect_key()</p>	<p>Compares array keys, and returns the matches</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Cat",1=&gt;"Dog",2=&gt;"Horse"); \$a2=array(2=&gt;"Bird",0=&gt;"Cat",4=&gt;"Fish"); print_r(array_intersect_key(\$a1,\$a2)); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Cat [2] =&gt; Horse )</p>
<p>array_intersect_uassoc()</p>	<p>Compares array keys and values, with an additional user-made function check, and returns the matches</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2){ if (\$v1=== \$v2){ return 0; } if (\$v1&gt;\$v2){ return 1; } else { return -1; } } \$a1=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); \$a2=array(3=&gt;"Dog",1=&gt;"Cat",5=&gt;"Horse"); print_r(array_intersect_uassoc(\$a1,\$a2,"myfunction"));</pre>

	<p>?&gt;</p> <p><b>Output</b>Array ( [1] =&gt; Cat )</p>
<p>array_intersect_ukey()</p>	<p>Compares array keys, with an additional user-made function check, and returns the matches</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2){ if (\$v1=== \$v2){ return 0;} if (\$v1&gt;\$v2){ return 1; } else{ return -1;} } \$a1=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse"); \$a2=array(3=&gt;"Rat",1=&gt;"Bird",5=&gt;"Monkey"); print_r(array_diff_ukey(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p><b>output</b>Array ( [0] =&gt; Dog [2] =&gt; Horse )</p>
<p>array_key_exists()</p>	<p>Checks if the specified key exists in the array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Dog","b"=&gt;"Cat"); if (array_key_exists("a",\$a)) { echo "Key exists!"; } else {</pre>

	<pre>echo "Key does not exist!"; } ?&gt;</pre> <p><b>Output</b>Key exists!</p>
array_keys()	<p>Returns all the keys of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Horse","b"=&gt;"Cat","c"=&gt;"Dog"); print_r(array_keys(\$a)); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; a [1] =&gt; b [2] =&gt; c )</p>
array_map()	<p>Sends each value of an array to a user-made function, which returns new values</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v) { if (\$v=="Dog"){ return "Fido"; } return \$v; } \$a=array("Horse","Dog","Cat"); print_r(array_map("myfunction",\$a)); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Horse [1] =&gt; Fido [2] =&gt; Cat )</p>
array_merge()	<p>Merges one or more arrays into one array</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array("a"=&gt;"Horse","b"=&gt;"Dog"); \$a2=array("c"=&gt;"Cow","b"=&gt;"Cat");</pre>



	<pre>print_r(array_merge(\$a1,\$a2)); ?&gt;</pre> <p>OutputArray ( [a] =&gt; Horse [b] =&gt; Cat [c] =&gt; Cow )</p>
array_merge_recursive()	<p>Merges one or more arrays into one array</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array("a"=&gt;"Horse","b"=&gt;"Dog"); \$a2=array("c"=&gt;"Cow","b"=&gt;"Cat"); print_r(array_merge_recursive(\$a1,\$a2)); ?&gt;</pre> <p><b>Output</b>Array ( [a] =&gt; Horse [b] =&gt; Array ( [0] =&gt; Dog [1] =&gt; Cat ) [c] =&gt; Cow )</p>
array_multisort()	<p>Sorts multiple or multi-dimensional arrays</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array("Dog","Cat"); \$a2=array("Fido","Missy"); array_multisort(\$a1,\$a2); print_r(\$a1); print_r(\$a2); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Cat [1] =&gt; Dog ) Array ( [0] =&gt; Missy [1] =&gt; Fido )</p>
array_pad()	<p>Inserts a specified number of items, with a specified value, to an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("Dog","Cat"); print_r(array_pad(\$a,5,0)); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Dog [1] =&gt; Cat [2] =&gt; 0 [3] =&gt; 0 [4] =&gt; 0 )</p>
array_pop()	<p>Deletes the last element of an array</p>

	<p><b>Example</b></p> <pre>&lt;?php \$a=array("Dog","Cat","Horse"); array_pop(\$a); print_r(\$a); ?&gt;</pre> <p><b>Output</b>Array ( [0] =&gt; Dog [1] =&gt; Cat )</p>
array_product()	<p>Calculates the product of the values in an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array(5,5); echo(array_product(\$a)); ?&gt;</pre> <p><b>Output</b>25</p>
array_push()	<p>Inserts one or more elements to the end of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("Dog","Cat"); array_push(\$a,"Horse","Bird"); print_r(\$a); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Dog [1] =&gt; Cat [2] =&gt; Horse [3] =&gt; Bird )</p>
array_rand()	<p>Returns one or more random keys from an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Dog","b"=&gt;"Cat","c"=&gt;"Horse"); print_r(array_rand(\$a,1)); ?&gt;</pre> <p><b>Output</b>b</p>
array_reduce()	<p>Returns an array as a string, using a user-defined function</p>

	<p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { return \$v1 . "-" . \$v2; } \$a=array("Dog","Cat","Horse"); print_r(array_reduce(\$a,"myfunction")); ?&gt;</pre> <p><b>Output-Dog-Cat-Horse</b></p>
<p>array_reverse()</p>	<p>Returns an array in the reverse order</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Dog","b"=&gt;"Cat","c"=&gt;"Horse"); print_r(array_reverse(\$a)); ?&gt;</pre> <p>OutputArray ( [c] =&gt; Horse [b] =&gt; Cat [a] =&gt; Dog )</p>
<p>array_search()</p>	<p>Searches an array for a given value and returns the key</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Dog","b"=&gt;"Cat","c"=&gt;"Horse"); echo array_search("Dog",\$a); ?&gt;</pre> <p><b>Outputa</b></p>
<p>array_shift()</p>	<p>Removes the first element from an array, and returns the value of the removed element</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Dog","b"=&gt;"Cat","c"=&gt;"Horse"); echo array_shift(\$a); print_r (\$a); ?&gt;</pre> <p>OutputDogArray ( [b] =&gt; Cat [c] =&gt; Horse )</p>

array_slice()	<p>Returns selected parts of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse",3=&gt;"Bird"); print_r(array_slice(\$a,1,2)); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Cat [1] =&gt; Horse )</p>
array_splice()	<p>Removes and replaces specified elements of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a1=array(0=&gt;"Dog",1=&gt;"Cat",2=&gt;"Horse",3=&gt;"Bird"); \$a2=array(0=&gt;"Tiger",1=&gt;"Lion"); array_splice(\$a1,0,2,\$a2); print_r(\$a1); ?&gt;</pre> <p>Output</p> <p>Array ( [0] =&gt; Tiger [1] =&gt; Lion [2] =&gt; Horse [3] =&gt; Bird )</p>
array_sum()	<p>Returns the sum of the values in an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array(0=&gt;"5",1=&gt;"15",2=&gt;"25"); echo array_sum(\$a); ?&gt;</pre> <p><b>Output</b> 45</p>
array_udiff()	<p>Compares array values in a user-made function and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; }</pre>

	<pre>\$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array(1=&gt;"Cat",2=&gt;"Dog",3=&gt;"Fish"); print_r(array_udiff(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p>OutputArray ( [c] =&gt; Horse )</p>
array_udiff_assoc()	<p>Compares array keys, and compares array values in a user-made function, and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; } \$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array("a"=&gt;"Cat","b"=&gt;"Horse","c"=&gt;"Dog"); print_r(array_udiff_assoc(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p>OutputArray ( [b] =&gt; Dog [c] =&gt; Horse )</p>
array_udiff_uassoc()	<p>Compares array keys and array values in user-made functions, and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction_key(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; }  function myfunction_value(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; }</pre>

	<pre>\$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Fish"); print_r(array_udiff_uassoc(\$a1,\$a2,"myfunction_key","myfunction_value")); ?&gt;</pre> <p>Output Array ( [c] =&gt; Horse )</p>
array_uintersect()	<p>Compares array values in a user-made function and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } if (\$v1 &gt; \$v2) return 1; { return -1; } return 1; } \$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array(1=&gt;"Cat",2=&gt;"Dog",3=&gt;"Fish"); print_r(array_uintersect(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p>Output Array ( [a] =&gt; Cat [b] =&gt; Dog )</p>
array_uintersect_assoc()	<p>Compares array keys, and compares array values in a user-made function, and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } }</pre>

	<pre>return 1; } \$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array("a"=&gt;"Cat","b"=&gt;"Horse","c"=&gt;"Dog"); print_r(array_intersect_assoc(\$a1,\$a2,"myfunction")); ?&gt;</pre> <p><b>Output</b></p> <p>Array ( [a] Cat )</p>
<p>array_intersect_uassoc( )</p>	<p>Compares array keys and array values in user-made functions, and returns an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunc_key(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; } function myfunc_value(\$v1,\$v2) { if (\$v1=== \$v2) { return 0; } return 1; } \$a1=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); \$a2=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Dog"); print_r(array_intersect_uassoc(\$a1,\$a2,"myfunc_key","myfunc_value")); ?&gt;</pre> <p>Output</p> <p>Array ( [a] =&gt; Cat [b] =&gt; Dog )</p>
<p>array_unique()</p>	<p>Removes duplicate values from an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Cat"); print_r(array_unique(\$a)); ?&gt;</pre> <p>Output</p>

	<p>Array ( [a] =&gt; Cat [b] =&gt; Dog )</p>
array_unshift()	<p>Adds one or more elements to the beginning of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Cat","b"=&gt;"Dog"); array_unshift(\$a,"Horse"); print_r(\$a); ?&gt;</pre> <p>Output</p> <p>Array ( [0] =&gt; Horse [a] =&gt; Cat [b] =&gt; Dog )</p>
array_values()	<p>Returns all the values of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); print_r(array_values(\$a)); ?&gt;</pre> <p>Output</p> <p>Array ( [0] =&gt; Cat [1] =&gt; Dog [2] =&gt; Horse )</p>
array_walk()	<p>Applies a user function to every member of an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$value,\$key) { echo "The key \$key has the value \$value&lt;br /&gt;"; } \$a=array("a"=&gt;"Cat","b"=&gt;"Dog","c"=&gt;"Horse"); array_walk(\$a,"myfunction"); ?&gt;</pre> <p>Output</p> <p>The key a has the value Cat  The key b has the value Dog  The key c has the value Horse</p>
array_walk_recursive()	<p>Applies a user function recursively to every member of an array</p> <p><b>Example</b></p> <pre>&lt;?php function myfunction(\$value,\$key)</pre>



	<pre>{ echo "The key \$key has the value \$value&lt;br /&gt;"; } \$a1=array("a"=&gt;"Cat","b"=&gt;"Dog"); \$a2=array(\$a1,"1"=&gt;"Bird","2"=&gt;"Horse"); array_walk_recursive(\$a2,"myfunction"); ?&gt;</pre> <p>Output</p> <p>The key a has the value Cat  The key b has the value Dog  The key 1 has the value Bird  The key 2 has the value Horse</p>
<p>arsort()</p>	<p>Sorts an array in reverse order and maintain index association</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog", "b" =&gt; "Cat", "c" =&gt; "Horse"); arsort(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [c] =&gt; Horse [a] =&gt; Dog [b] =&gt; Cat )</p>
<p>asort()</p>	<p>Sorts an array and maintain index association</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog", "b" =&gt; "Cat", "c" =&gt; "Horse"); asort(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [b] =&gt; Cat [a] =&gt; Dog [c] =&gt; Horse )</p>
<p>compact()</p>	<p>Create array containing variables and their values</p> <p><b>Example</b></p> <pre>&lt;?php \$firstname = "Peter"; \$lastname = "Griffin"; \$age = "38"; \$result = compact("firstname", "lastname", "age"); print_r(\$result); ?&gt;</pre>

	<p>OutputArray ( [firstname] =&gt; Peter [lastname] =&gt; Griffin [age] =&gt; 38 )</p>
count()	<p>Counts elements in an array, or properties in an object</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); \$result = count(\$people); echo \$result; ?&gt;</pre> <p>Output4</p>
current()	<p>Returns the current element in an array</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current(\$people) . "&lt;br /&gt;"; ?&gt;</pre> <p>OutputPeter</p>
each()	<p>Returns the current key and value pair from an array</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); print_r (each(\$people)); ?&gt;</pre> <p>Output</p> <p>Array ( [1] =&gt; Peter [value] =&gt; Peter [0] =&gt; 0 [key] =&gt; 0 )</p>
end()	<p>Sets the internal pointer of an array to its last element</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current(\$people) . "&lt;br /&gt;"; echo end(\$people); ?&gt;</pre> <p>Output</p> <p>Peter Cleveland</p>

<p>extract()</p>	<p>Imports variables into the current symbol table from an array</p> <p><b>Example</b></p> <pre>&lt;?php \$a = 'Original'; \$my_array = array("a" =&gt; "Cat","b" =&gt; "Dog", "c" =&gt; "Horse"); extract(\$my_array); echo "\\$a = \$a; \\$b = \$b; \\$c = \$c"; ?&gt;</pre> <p>Output\$a = Cat; \$b = Dog; \$c = Horse</p>
<p>in_array()</p>	<p>Checks if a specified value exists in an array</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); if (in_array("Glenn",\$people)){     echo "Match found"; } else{     echo "Match not found"; } ?&gt;</pre> <p>OutputMatch found</p>
<p>key()</p>	<p>Fetches a key from an array</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo "The key from the current position is: " . key(\$people); ?&gt;</pre> <p>OutputThe key from the current position is: 0</p>
<p>krsort()</p>	<p>Sorts an array by key in reverse order</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog", "b" =&gt; "Cat","c" =&gt; "Horse"); krsort(\$my_array); print_r(\$my_array); ?&gt;</pre>

	<p>OutputArray ( [c] =&gt; Horse [b] =&gt; Cat [a] =&gt; Dog )</p>
ksort()	<p>Sorts an array by key</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt;"Dog", "b" =&gt; "Cat", "c" =&gt;"Horse"); ksort(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [a] =&gt; Dog [b] =&gt; Cat [c] =&gt; Horse )</p>
list()	<p>Assigns variables as if they were an array</p> <p><b>Example</b></p> <pre>&lt;?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."; ?&gt;</pre> <p>OutputI have several animals, a Dog, a Cat and a Horse.</p>
natcasesort()	<p>Sorts an array using a case insensitive "natural order" algorithm</p> <p><b>Example</b></p> <pre>&lt;?php \$temp_files = array("temp15.txt","Temp10.txt", "temp1.txt","Temp22.txt","temp2.txt"); natsort(\$temp_files); echo "Natural order: "; print_r(\$temp_files); echo "&lt;br /&gt;"; natcasesort(\$temp_files); echo "Natural order case insensitive: "; print_r(\$temp_files); ?&gt;</pre> <p>OutputNatural order: Array ( [1] =&gt; Temp10.txt [3] =&gt; Temp22.txt [2] =&gt; temp1.txt [4] =&gt; temp2.txt [0] =&gt; temp15.txt )  Natural order case insensitive: Array ( [2] =&gt; temp1.txt [4] =&gt; temp2.txt [1] =&gt; Temp10.txt [0] =&gt; temp15.txt [3] =&gt; Temp22.txt )</p>
natsort()	<p>Sorts an array using a "natural order" algorithm</p> <p><b>Example</b></p>

	<pre>&lt;?php \$temp_files = array("temp15.txt","temp10.txt", "temp1.txt","temp22.txt","temp2.txt");  sort(\$temp_files); echo "Standard sorting: "; print_r(\$temp_files); echo "&lt;br /&gt;";  natsort(\$temp_files); echo "Natural order: "; print_r(\$temp_files); ?&gt;</pre> <p>OutputStandard sorting: Array ( [0] =&gt; temp1.txt [1] =&gt; temp10.txt [2] =&gt; temp15.txt [3] =&gt; temp2.txt [4] =&gt; temp22.txt )  Natural order: Array ( [0] =&gt; temp1.txt [3] =&gt; temp2.txt [1] =&gt; temp10.txt [2] =&gt; temp15.txt [4] =&gt; temp22.txt )</p>
next()	<p>Advance the internal array pointer of an array</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current(\$people) . "&lt;br /&gt;"; echo next(\$people); ?&gt;</pre> <p>OutputPeter  Joe</p>
pos()	<p>Alias of current()</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo pos(\$people) . "&lt;br /&gt;"; ?&gt;</pre> <p>OutputPeter</p>
prev()	<p>Rewinds the internal array pointer</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current(\$people) . "&lt;br /&gt;";</pre>

	<pre>echo next(\$people) . "&lt;br /&gt;"; echo prev(\$people); ?&gt;</pre> <p>Output:Peter</p> <p style="padding-left: 40px;">Joe</p> <p style="padding-left: 40px;">Peter</p>
range()	<p>Creates an array containing a range of elements</p> <p><b>Example</b></p> <pre>&lt;?php \$number = range(0,5); print_r (\$number); ?&gt;</pre> <p>OutputArray ( [0] =&gt; 0 [1] =&gt; 1 [2] =&gt; 2 [3] =&gt; 3 [4] =&gt; 4 [5] =&gt; 5 )</p>
reset()	<p>Sets the internal pointer of an array to its first element</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current(\$people) . "&lt;br /&gt;"; echo next(\$people) . "&lt;br /&gt;"; echo reset(\$people); ?&gt;</pre> <p>OutputPeter</p> <p style="padding-left: 40px;">Joe</p> <p style="padding-left: 40px;">Peter</p>
rsort()	<p>Sorts an array in reverse order</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog", "b" =&gt; "Cat", "c" =&gt; "Horse"); rsort(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Horse [1] =&gt; Dog [2] =&gt; Cat )</p>
shuffle()	<p>Shuffles an array</p>

	<p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog", "b" =&gt; "Cat", "c" =&gt; "Horse"); shuffle(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Horse [1] =&gt; Cat [2] =&gt; Dog )</p>
sizeof()	<p>Alias of count()</p> <p><b>Example</b></p> <pre>&lt;?php \$people = array("Peter", "Joe", "Glenn", "Cleveland"); \$result = sizeof(\$people); echo \$result; ?&gt;</pre> <p>Output 4</p>
sort()	<p>Sorts an array</p> <p><b>Example</b></p> <pre>&lt;?php \$my_array = array("a" =&gt; "Dog","b" =&gt; "Cat", "c" =&gt; "Horse"); sort(\$my_array); print_r(\$my_array); ?&gt;</pre> <p>OutputArray ( [0] =&gt; Cat [1] =&gt; Dog [2] =&gt; Horse )</p>
uasort()	<p>Sorts an array with a user-defined function and maintain index association</p> <p><b>Example</b></p> <pre>&lt;?php function my_sort(\$a, \$b) { if (\$a == \$b) return 0; return (\$a &gt; \$b) ? -1 : 1; } \$people = array("Swanson" =&gt; "Joe", "Griffin" =&gt; "Peter", "Quagmire" =&gt; "Glenn", "swanson" =&gt; "joe", "griffin" =&gt; "peter", "quagmire" =&gt; "glenn"); uasort(\$people, "my_sort");</pre>

	<pre>print_r (\$people); ?&gt;</pre> <p>OutputArray ( [griffin] =&gt; peter [swanson] =&gt; joe [quagmire] =&gt; glenn [Griffin] =&gt; Peter [Swanson] =&gt; Joe [Quagmire] =&gt; Glenn )</p>
uksort()	<p>Sorts an array by keys using a user-defined function</p> <p><b>Example</b></p> <pre>&lt;?php function my_sort(\$a, \$b){     if (\$a == \$b) return 0;     return (\$a &gt; \$b) ? -1 : 1; } \$people = array("Swanson" =&gt; "Joe", "Griffin" =&gt; "Peter", "Quagmire" =&gt; "Glenn", "swanson" =&gt; "joe", "griffin" =&gt; "peter", "quagmire" =&gt; "glenn"); uksort(\$people, "my_sort"); print_r (\$people); ?&gt;</pre> <p>OutputArray ( [swanson] =&gt; joe [quagmire] =&gt; glenn [griffin] =&gt; peter [Swanson] =&gt; Joe [Quagmire] =&gt; Glenn [Griffin] =&gt; Peter )</p>
usort()	<p>Sorts an array by values using a user-defined function</p> <p><b>Example</b></p> <pre>&lt;?php function my_sort(\$a, \$b){     if (\$a == \$b) return 0;     return (\$a &gt; \$b) ? -1 : 1; } \$arr = array("Peter", "glenn","Cleveland", "peter","cleveland", "Glenn"); usort(\$arr, "my_sort"); print_r (\$arr); ?&gt;</pre> <p>OutputArray ( [0] =&gt; peter [1] =&gt; glenn [2] =&gt; cleveland [3] =&gt; Peter [4] =&gt; Glenn [5] =&gt; Cleveland )</p>