Focus search box

uasort »
« sizeof

- Manual de PHP
- Referencia de funciones
- Extensiones relacionadas con variable y tipo
- Arrays
- Funciones de Arrays

Change language: Spanish

Submit a Pull Request Report a Bug

sort

(PHP 4, PHP 5, PHP 7, PHP 8)

sort — Ordena un array

Descripción ¶

sort(array `&$array`, int `$sort_flags` = SORT_REGULAR): bool

Esta función ordena un array. Los elementos estarán ordenados de menor a mayor cuando la función haya terminado.

> **Nota**:
>
> Si dos miembros se comparan como iguales, su orden relativo en el array oredenado será indefinido.

Parámetros ¶

`array`

> El array de entrada.

`sort_flags`

> El segundo parámetro opcional `sort_flags` puede ser usado para modificar el modo de ordenación usando estos valores:
>
> Tipos de ordenación:
>
> - **`SORT_REGULAR`** - compara elementos normalmente (no cambia los tipos)
> - **`SORT_NUMERIC`** - compara elementos de forma numérica
> - **`SORT_STRING`** - compara elementos como cadenas
> - **`SORT_LOCALE_STRING`** - compara elementos como cadenas, basándose en la configuración regional en uso. Utiliza la configuración regional, la cual puede cambiarse usando setlocale().
> - **`SORT_NATURAL`** - compara elementos como cadenas usando el "orden natural" de la misma forma que natsort().
> - **`SORT_FLAG_CASE`** - se puede combinar (OR a nivel de bits) con **`SORT_STRING`** o **`SORT_NATURAL`** para ordenar cadenas de forma insensible a mayúsculas/minúsculas.

Valores devueltos ¶

Devuelve **`true`** en caso de éxito o **`false`** en caso de error.

Historial de cambios ¶

Versión	Descripción
5.4.0	Se añadió el soporte para **`SORT_NATURAL`** y **`SORT_FLAG_CASE`** como `sort_flags`
5.0.2	Se añadió **`SORT_LOCALE_STRING`**

Ejemplos ¶

Ejemplo #1 Ejemplo de sort()

```php
<?php

$frutas = array("limón", "naranja", "banana", "albaricoque");
sort($frutas);
foreach ($frutas as $clave => $valor) {
    echo "frutas[" . $clave . "] = " . $valor . "\n";
}

?>
```

El resultado del ejemplo sería:

```
frutas[0] = albaricoque
frutas[1] = banana
frutas[2] = limón
frutas[3] = naranja
```

Las frutas han sido ordenadas en orden alfabético.

Ejemplo #2 Ejemplo de sort() usando la ordenación insensible a mayúsculas/minúsculas natural

```php
<?php

$frutas = array(
    "Naranja1", "naranja2", "Naranja3", "naranja20"
);
sort($frutas, SORT_NATURAL | SORT_FLAG_CASE);
foreach ($frutas as $clave => $valor) {
    echo "frutas[" . $clave . "] = " . $valor . "\n";
}

?>
```

El resultado del ejemplo sería:

```
frutas[0] = Naranja1
frutas[1] = naranja2
frutas[2] = Naranja3
frutas[3] = naranja20
```

Las frutas han sido ordenadas de la misma forma que natcasesort().

Notas ¶

> **Nota**: Esta función asigna nuevas clave a los elemenos del `array`. Eliminará cualquier clave existente que haya sido asignada, en lugar de reordenar las claves.

Nota: Como la mayoría de funciones de ordenación de PHP, **sort()** utiliza una implementación de » Quicksort. El pivote es elegido en la mitad de la partición resultando en un tiempo óptimo para los arrays ya ordenados. Aunque esto es un detalle de implementación con el que no debería contar.

Advertencia

Se ha de tener cuidado cuando se ordenen arrays con valores de tipos mixtos ya que **sort()** puede producir resultados impredecibles.

Ver también ¶

- asort() - Ordena un array y mantiene la asociación de índices
- comparación de funciones de orden de arrays

+ add a note

User Contributed Notes 36 notes

up
down
207
phpdotnet at m4tt dot co dot uk ¶
12 years ago

```
Simple function to sort an array by a specific key. Maintains index association.

<?php

function array_sort($array, $on, $order=SORT_ASC)
{
    $new_array = array();
    $sortable_array = array();

    if (count($array) > 0) {
        foreach ($array as $k => $v) {
            if (is_array($v)) {
                foreach ($v as $k2 => $v2) {
                    if ($k2 == $on) {
                        $sortable_array[$k] = $v2;
                    }
                }
            } else {
                $sortable_array[$k] = $v;
            }
        }

        switch ($order) {
            case SORT_ASC:
                asort($sortable_array);
            break;
            case SORT_DESC:
                arsort($sortable_array);
            break;
        }

        foreach ($sortable_array as $k => $v) {
            $new_array[$k] = $array[$k];
```

```
        }
    }

    return $new_array;
}

$people = array(
    12345 => array(
        'id' => 12345,
        'first_name' => 'Joe',
        'surname' => 'Bloggs',
        'age' => 23,
        'sex' => 'm'
    ),
    12346 => array(
        'id' => 12346,
        'first_name' => 'Adam',
        'surname' => 'Smith',
        'age' => 18,
        'sex' => 'm'
    ),
    12347 => array(
        'id' => 12347,
        'first_name' => 'Amy',
        'surname' => 'Jones',
        'age' => 21,
        'sex' => 'f'
    )
);

print_r(array_sort($people, 'age', SORT_DESC)); // Sort by oldest first
print_r(array_sort($people, 'surname', SORT_ASC)); // Sort by surname

/*
Array
(
    [12345] => Array
        (
            [id] => 12345
            [first_name] => Joe
            [surname] => Bloggs
            [age] => 23
            [sex] => m
        )

    [12347] => Array
        (
            [id] => 12347
            [first_name] => Amy
            [surname] => Jones
            [age] => 21
            [sex] => f
        )

    [12346] => Array
        (
            [id] => 12346
            [first_name] => Adam
```

```
            [surname] => Smith
            [age] => 18
            [sex] => m
        )

)
Array
(
    [12345] => Array
        (
            [id] => 12345
            [first_name] => Joe
            [surname] => Bloggs
            [age] => 23
            [sex] => m
        )

    [12347] => Array
        (
            [id] => 12347
            [first_name] => Amy
            [surname] => Jones
            [age] => 21
            [sex] => f
        )

    [12346] => Array
        (
            [id] => 12346
            [first_name] => Adam
            [surname] => Smith
            [age] => 18
            [sex] => m
        )

)
*/

?>
```

up
down
1
aminkhoshzahmat at gmail dot com ¶
2 years ago

```
Let's say we have a list of names, and it is not sorted.

<?php

$names = array('Amin', 'amir', 'sarah', 'Somayeh', 'armita', 'Armin');

sort($names); // simple alphabetical sort
print_r($names);
?>
Result is :
Array
(
    [0] => Amin
    [1] => Armin
```

```
    [2] => Somayeh // actually it's not sort alphabetically from here!
    [3] => amir         // comparison is based on ASCII values.
    [4] => armita
    [5] => sarah
)

If you want to sort alphabeticaly no matter it is upper or lower case:

<?php

sort($names, SORT_STRING | SORT_FLAG_CASE);
print_r($names);
?>

Result is:
Array
(
    [0] => Amin
    [1] => amir
    [2] => Armin
    [3] => armita
    [4] => sarah
    [5] => Somayeh
)
```

up
down
6

ajanata at gmail dot com ¶
11 years ago

This took me longer than it should have to figure out, but if you want the behavior of sort($array, SORT_STRING) (that is, re-indexing the array unlike natcasesort) in a case-insensitive manner, it is a simple matter of doing usort($array, strcasecmp).

up
down
5

joris at mangrove dot nl ¶
15 years ago

Commenting on note http://www.php.net/manual/en/function.sort.php#62311 :

Sorting an array of objects will not always yield the results you desire.

As pointed out correctly in the note above, sort() sorts the array by value of the first member variable. However, you can not always assume the order of your member variables! You must take into account your class hierarchy!

By default, PHP places the inherited member variables on top, meaning your first member variable is NOT the first variable in your class definition!
However, if you use code analyzers or a compile cache, things can be very different. E.g., in eAccelerator, the inherited member variables are at the end, meaning you get different sort results with caching on or off.

Conclusion:
Never use sort on arrays with values of a type other than scalar or array.

up
down
9

Walter Tross ¶
10 years ago

unless you specify the second argument, "regular" comparisons will be used. I quote from the page on comparison operators:
"If you compare a number with a string or the comparison involves numerical strings, then each string is converted to a number and the comparison performed numerically."
What this means is that "10" < "1a", and "1a" < "2", but "10" > "2". In other words, regular PHP string comparisons are not transitive.
This implies that the output of sort() can in rare cases depend on the order of the input array:

```
<?php
function echo_sorted($a)
{
   echo "{$a[0]} {$a[1]} {$a[2]}";
   sort($a);
   echo " => {$a[0]} {$a[1]} {$a[2]}\n";
}
// on PHP 5.2.6:
echo_sorted(array( "10", "1a", "2")); // => 10 1a 2
echo_sorted(array( "10", "2", "1a")); // => 1a 2 10
echo_sorted(array( "1a", "10", "2")); // => 2 10 1a
echo_sorted(array( "1a", "2", "10")); // => 1a 2 10
echo_sorted(array( "2", "10", "1a")); // => 2 10 1a
echo_sorted(array( "2", "1a", "10")); // => 10 1a 2
?>
```

up
down
5

danm68 at gmail dot com ¶

13 years ago

sort() used with strings doesn't sort just alphabetically. It sorts all upper-case strings alphabetically first and then sorts lower-case strings alphabetically second.
Just in case anyone was as confused as I was and I've never seen this mentioned anywhere.

up
down
2

peek at mailandnews dot com ¶

21 years ago

I ran into the same problem with case insensitive sorting. Actually I think there should be a SORT_STRING_CASE flag but I tried the following:

```
usort($listing, 'strcasecmp');
```

This didn't work (why not?), but you can do a proper case insensitive sort like this:

```
usort($listing, create_function('$a,$b','return strcasecmp($a,$b);'));
```

up
down
2

eriewave at hotmail dot com ¶

12 years ago

If you need to sort an array containing some equivalent values and you want the equivalents to end up next to each other in the overall order (similar to a MySQL's ORDER BY output), rather than breaking the function, do this:

```
<?php

sort($array, ksort($array))

?>
```

-When the sort() function finds two equivalents, it will sort them arbitrarily by their key #'s as a second parameter.

-Dirk

up
down
1

williamprogphp at[pleaseNOTSPAM] yahoo d ¶

8 years ago

In order to make some multidimensional quick sort implementation, take advantage of this stuff

```
<?php
        function quickSortMultiDimensional($array, $chave) {
            if( count( $array ) < 2 ) {
                return $array;
            }
            $left = $right = array( );
            reset( $array );
            $pivot_key    = key( $array );
            $pivot    = array_shift( $array );
            foreach( $array as $k => $v ) {
                if( $v[$chave] < $pivot[$chave] )
                        $left[$k][$chave] = $v[$chave];
                else
                        $right[$k][$chave] = $v[$chave];
            }
            return array_merge(
                                    quickSortMultiDimensional($left, $chave),
                                    array($pivot_key => $pivot),
                                    quickSortMultiDimensional($right, $chave)
            );
        }
?>
```

I make it using the idea from pageconfig dot com

tks for viewing

up
down
1

alishahnovin at hotmail dot com ¶

15 years ago

I had a multidimensional array, which needed to be sorted by one of the keys. This is what I came up with...

```
<?php
function msort($array, $id="id") {
        $temp_array = array();
        while(count($array)>0) {
            $lowest_id = 0;
            $index=0;
            foreach ($array as $item) {
                if ($item[$id]<$array[$lowest_id][$id]) {
                    $lowest_id = $index;
                }
                $index++;
            }
            $temp_array[] = $array[$lowest_id];
```

```
            $array = array_merge(array_slice($array, 0,$lowest_id), array_slice($array,
$lowest_id+1));
        }
        return $temp_array;
    }
?>

Ex:

<?php

//oh no, this is not in the ordered by id!!
$data[] = array("item"=>"item 4", "id"=>4);
$data[] = array("item"=>"item 1", "id"=>1);
$data[] = array("item"=>"item 3", "id"=>3);
$data[] = array("item"=>"item 2", "id"=>2);

var_dump(  msort($data)  ); //just msort it!

/* outputs

array
  0 =>
    array
      'item' => 'item 1' (length=6)
      'id' => 1
  1 =>
    array
      'item' => 'item 2' (length=6)
      'id' => 2
  2 =>
    array
      'item' => 'item 3' (length=6)
      'id' => 3
  3 =>
    array
      'item' => 'item 4' (length=6)
      'id' => 4

*/

?>
```

up
down
1

g8z at yahoo dot com ¶

16 years ago

```
<?php
/**
This sort function allows you to sort an associative array while "sticking" some fields.

$sticky_fields = an array of fields that should not be re-sorted. This is a method of achieving
sub-sorts within contiguous groups of records that have common data in some fields.

Courtesy of the $5 Script Archive: http://www.tufat.com
**/

define( 'ASC_AZ', 1000 );
```

```
define( 'DESC_AZ', 1001 );
define( 'ASC_NUM', 1002 );
define( 'DESC_NUM', 1003 );

function stickysort( $arr, $field, $sort_type, $sticky_fields = array() ) {
    $i = 0;
    foreach ($arr as $value) {
        $is_contiguous = true;
        if(!empty($grouped_arr)) {
            $last_value = end($grouped_arr[$i]);

            if(!($sticky_fields == array())) {
                foreach ($sticky_fields as $sticky_field) {
                    if ($value[$sticky_field] <> $last_value[$sticky_field]) {
                        $is_contiguous = false;
                        break;
                    }
                }
            }
        }
        if ($is_contiguous)
            $grouped_arr[$i][] = $value;
        else
            $grouped_arr[++$i][] = $value;
    }
    $code = '';
    switch($sort_type) {
        case ASC_AZ:
            $code .= 'return strcasecmp($a["'.$field.'"], $b["'.$field.'"]);';
            break;
        case DESC_AZ:
            $code .= 'return (-1*strcasecmp($a["'.$field.'"], $b["'.$field.'"]));';
            break;
        case ASC_NUM:
            $code .= 'return ($a["'.$field.'"] - $b["'.$field.'"]);';
            break;
        case DESC_NUM:
            $code .= 'return ($b["'.$field.'"] - $a["'.$field.'"]);';
            break;
    }

    $compare = create_function('$a, $b', $code);

    foreach($grouped_arr as $grouped_arr_key=>$grouped_arr_value)
        usort ( $grouped_arr[$grouped_arr_key], $compare );

    $arr = array();
    foreach($grouped_arr as $grouped_arr_key=>$grouped_arr_value)
        foreach($grouped_arr[$grouped_arr_key] as $grouped_arr_arr_key=>$grouped_arr_arr_value)
            $arr[] = $grouped_arr[$grouped_arr_key][$grouped_arr_arr_key];

    return $arr;
}
?>
```

up
down
1
arjan321 at hotmail dot com ¶

19 years ago

Ik you want to sort case insensitive, use the natcasesort()

up
down
1

petr dot biza at gmail dot com ¶

13 years ago

Here is a function to sort an array by the key of his sub-array with keep key in top level.

```
<?php
function sksort(&$array, $subkey="id", $sort_descending=false, $keep_keys_in_sub = false) {
    $temp_array = $array;

    foreach ($temp_array as $key => &$value) {

      $sort = array();
      foreach ($value as $index => $val) {
          $sort[$index] = $val[$subkey];
      }

      asort($sort);

      $keys = array_keys($sort);
      $newValue = array();
      foreach ($keys as $index) {
        if($keep_keys_in_sub)
            $newValue[$index] = $value[$index];
        else
            $newValue[] = $value[$index];
      }

      if($sort_descending)
        $value = array_reverse($newValue, $keep_keys_in_sub);
      else
        $value = $newValue;
    }

    $array = $temp_array;
  }
?>
```

up
down
1

matpatnik at hotmail dot com ¶

14 years ago

This function will sort entity letters eg:é

I hope that help someone

```
function sort_entity($array) {
    $total = count($array);
    for ($i=0;$i<$total;$i++) {
        if ($array[$i]{0} == '&') {
            $array[$i] = $array[$i]{1}.$array[$i];
        } else {
            $array[$i] = $array[$i]{0}.$array[$i];
        }
    }
```

```
    sort($array);

    for ($i=0;$i<$total;$i++) {
        $array[$i] = substr($array[$i],1);
    }

    return $array;
}
```

up
down
0
Md. Abutaleb ¶
2 years ago

```
<?php
/*
As I found the sort() function normally works as ascending order based on the following priority :
1. NULL
2. Empty
3. Boolean FALSE
4. String
5. Float
6. Int
7. Array
8. Object

Consider the following array:
*/

$a = ['fruit'=> 'apple', 'A' => 10, 20, 5, 2.5, 5=>'A new value', 'last' => 'value', TRUE, NULL,
"", FALSE, array(), new StdClass];
sort($a);
var_dump($a);

#The output is:

array(13) {
  [0]=>NULL
  [1]=> string(0) ""
  [2]=>bool(false)
  [3]=>string(11) "A new value"
  [4]=>string(5) "apple"
  [5]=>string(5) "value"
  [6]=> float(2.5)
  [7]=> int(5)
  [8]=>int(10)
  [9]=>int(20)
  [10]=>array(0) { }
  [11]=> bool(true)
  [12]=>object(stdClass)#1 (0) {}
}

//Hope it will remove your confusion when you're sorting an array with mix type data.
?>
```

up
down
0
r at rcse dot de ¶
3 years ago

Here is no word about sorting UTF-8 strings by any collation. This should not be so uncommon?

up
down
0

Abhishek Banerjee ¶
6 years ago

EDIT: To the original note by "phpdotnet at m4tt dot co dot uk"
Use array_push instead of $new_array[$k] for some reason it was
giving me string indexes.

Simple function to sort an array by a specific key. Maintains index association.

```
<?php

function array_sort($array, $on, $order=SORT_ASC)
{
    $new_array = array();
    $sortable_array = array();

    if (count($array) > 0) {
        foreach ($array as $k => $v) {
            if (is_array($v)) {
                foreach ($v as $k2 => $v2) {
                    if ($k2 == $on) {
                        $sortable_array[$k] = $v2;
                    }
                }
            } else {
                $sortable_array[$k] = $v;
            }
        }

        switch ($order) {
            case SORT_ASC:
                asort($sortable_array);
            break;
            case SORT_DESC:
                arsort($sortable_array);
            break;
        }

        foreach ($sortable_array as $k => $v) {
            array_push($new_array, $array[$k]);
        }
    }

    return $new_array;
}

$people = array(
    12345 => array(
        'id' => 12345,
        'first_name' => 'Joe',
        'surname' => 'Bloggs',
        'age' => 23,
        'sex' => 'm'
    ),
    12346 => array(
```

```
        'id' => 12346,
        'first_name' => 'Adam',
        'surname' => 'Smith',
        'age' => 18,
        'sex' => 'm'
    ),
    12347 => array(
        'id' => 12347,
        'first_name' => 'Amy',
        'surname' => 'Jones',
        'age' => 21,
        'sex' => 'f'
    )
);

print_r(array_sort($people, 'age', SORT_DESC)); // Sort by oldest first
print_r(array_sort($people, 'surname', SORT_ASC)); // Sort by surname

/*
Array
(
    [12345] => Array
        (
            [id] => 12345
            [first_name] => Joe
            [surname] => Bloggs
            [age] => 23
            [sex] => m
        )

    [12347] => Array
        (
            [id] => 12347
            [first_name] => Amy
            [surname] => Jones
            [age] => 21
            [sex] => f
        )

    [12346] => Array
        (
            [id] => 12346
            [first_name] => Adam
            [surname] => Smith
            [age] => 18
            [sex] => m
        )

)
Array
(
    [12345] => Array
        (
            [id] => 12345
            [first_name] => Joe
            [surname] => Bloggs
            [age] => 23
            [sex] => m
```

```
        )

    [12347] => Array
        (
            [id] => 12347
            [first_name] => Amy
            [surname] => Jones
            [age] => 21
            [sex] => f
        )

    [12346] => Array
        (
            [id] => 12346
            [first_name] => Adam
            [surname] => Smith
            [age] => 18
            [sex] => m
        )

)
*/

?>
```

up
down
0
aditycse at gmail dot com ¶
7 years ago

```
/*
* Name : Aditya Mehrotra
* Email: aditycse@gmail.com
*/
//Example for sorting by values for an alphanumeric array also having case-sensitive data
$exampleArray1 = $exampleArray2 = array(
    0 => 'example1',
    1 => 'Example10',
    2 => 'example12',
    3 => 'Example2',
    4 => 'example3',
    5 => 'EXAMPLE10',
    6 => 'example10'
);

//default sorting
asort($exampleArray1);

// alphanumeric with case-sensitive data sorting by values
asort($exampleArray2, SORT_STRING | SORT_FLAG_CASE | SORT_NATURAL);

//output of defaut sorting
print_r($exampleArray1);
/*
* output of default sorting
  Array
  (
  [5] => EXAMPLE10
  [1] => Example10
```

```
  [3] => Example2
  [0] => example1
  [6] => example10
  [2] => example12
  [4] => example3
  )
*/

print_r($exampleArray2);
/*
* output of alphanumeric with case-sensitive data sorting by values
Array
(
    [0] => example1
    [3] => Example2
    [4] => example3
    [5] => EXAMPLE10
    [1] => Example10
    [6] => example10
    [2] => example12
)
*/
```

up
down
0

me[at]szczepan[dot]info ¶
9 years ago

Sorting the keys, but keep the values in order is not possible by just ordering, because it would result in a new array. This is also the solution: Create a new array

```
<?php
$a = array(9=>"a",8=>"c",5=>"d");

$keys = array_keys($a);
sort($keys);
$result = array_combine($keys, array_values($a));

//Result : array(5=>"a",8=>"c",9=>"d");
?>
```

up
down
1

james at miicro dot net ¶
17 years ago

It's useful to know that if you're using this function on a multidimensional array, php will sort the first key, then the second and so on. This is similar to being able to use SQL to order by field1, field2 etc.

So:

```
Array (
[0] => Array ( [category] => work [name] => Smith )
[1] => Array ( [category] => play [name] => Johnson )
[2] => Array ( [category] => work [name] => Berger )
)

will become:
```

```
Array (
[0] => Array ( [category] => play [name] => Johnson )
[1] => Array ( [category] => work [name] => Berger )
[2] => Array ( [category] => work [name] => Smith )
)
```

Hope it helps someone.

up
down
0

alex dot hristov dot 88 at gmail dot com ¶

11 years ago

As some people have mentioned before sorting a multidimentional array can be a bit tricky. it took me quite a while to get it going but it works as a charm:

```
<?php
//$order has to be either asc or desc
function sortmulti ($array, $index, $order, $natsort=FALSE, $case_sensitive=FALSE) {
        if(is_array($array) && count($array)>0) {
            foreach(array_keys($array) as $key)
            $temp[$key]=$array[$key][$index];
            if(!$natsort) {
                if ($order=='asc')
                    asort($temp);
                else
                    arsort($temp);
            }
            else
            {
                if ($case_sensitive===true)
                    natsort($temp);
                else
                    natcasesort($temp);
            if($order!='asc')
                $temp=array_reverse($temp,TRUE);
            }
            foreach(array_keys($temp) as $key)
                if (is_numeric($key))
                    $sorted[]=$array[$key];
                else
                    $sorted[$key]=$array[$key];
            return $sorted;
        }
    return $sorted;
}
?>
```

up
down
0

stepmuel at ee dot ethz dot ch ¶

13 years ago

A little shorter way to sort an array of objects; with a callback function.

```
<?php
function objSort(&$objArray,$indexFunction,$sort_flags=0) {
    $indices = array();
    foreach($objArray as $obj) {
        $indeces[] = $indexFunction($obj);
```

```
    }
    return array_multisort($indeces,$objArray,$sort_flags);
}

function getIndex($obj) {
    return $obj->getPosition();
}

objSort($objArray,'getIndex');
?>
```

up
down
0

alex [at] vkpb [dot] com ¶
15 years ago

Sorting of an array by a method of inserts.

```
<?


       function sortByField($multArray,$sortField,$desc=true){
            $tmpKey='';
            $ResArray=array();

            $maIndex=array_keys($multArray);
            $maSize=count($multArray)-1;

            for($i=0; $i < $maSize ; $i++) {

               $minElement=$i;
               $tempMin=$multArray[$maIndex[$i]][$sortField];
               $tmpKey=$maIndex[$i];

                for($j=$i+1; $j <= $maSize; $j++)
                  if($multArray[$maIndex[$j]][$sortField] < $tempMin ) {
                     $minElement=$j;
                     $tmpKey=$maIndex[$j];
                     $tempMin=$multArray[$maIndex[$j]][$sortField];

                  }
                  $maIndex[$minElement]=$maIndex[$i];
                  $maIndex[$i]=$tmpKey;
            }

           if($desc)
               for($j=0;$j<=$maSize;$j++)
                  $ResArray[$maIndex[$j]]=$multArray[$maIndex[$j]];
           else
              for($j=$maSize;$j>=0;$j--)
                  $ResArray[$maIndex[$j]]=$multArray[$maIndex[$j]];

           return $ResArray;
       }

// make array
$array['aaa']=array("name"=>"vasia","order"=>1);
$array['bbb']=array("name"=>"petia","order"=>2);
$array['ccc']=array("name"=>"kolia","order"=>3);
```

```
$array['ddd']=array("name"=>"zenia","order"=>4);

// set sort
$SortOrder=0; // desc by default , 1- asc

var_dump(sortByField($array,'order',$SortOrder));

array
  'ddd' =>
    array
      'name' => 'zenia' (length=5)
      'order' => 4
  'aaa' =>
    array
      'name' => 'vasia' (length=5)
      'order' => 1
  'bbb' =>
    array
      'name' => 'petia' (length=5)
      'order' => 2
  'ccc' =>
    array
      'name' => 'kolia' (length=5)
      'order' => 3

?>
```

up
down
0

g8z at yahoo dot com ¶

16 years ago

```
<?php
/**
This sort function allows you to sort an associative array while "sticking" some fields.

$sticky_fields = an array of fields that should not be re-sorted. This is a method of achieving
sub-sorts within contiguous groups of records that have common data in some fields.

For example:

$a = array();

$a []= array(
    'name'         => 'Sam',
    'age'         => 23,
    'hire_date'    => '2004-01-01'
);
$a []= array(
    'name'        => 'Sam',
    'age'        => 44,
    'hire_date'    => '2003-03-23'
);
$a []= array(
    'name'        => 'Jenny',
    'age'        => 20,
    'hire_date' => '2000-12-31'
);
$a []= array(
```

```
    'name'        => 'Samantha',
    'age'        => 50,
    'hire_date' => '2000-12-14'
);

$sticky_fields = array( 'name' );
print_r( stickysort( $a, 'age', DESC_NUM, $sticky_fields ) );

OUTPUT:

Array
(
    [0] => Array
        (
            [name] => Sam
            [age] => 44
            [hire_date] => 2003-03-23
        )
    [1] => Array
        (
            [name] => Sam
            [age] => 23
            [hire_date] => 2004-01-01
        )
    [2] => Array
        (
            [name] => Jenny
            [age] => 20
            [hire_date] => 2000-12-31
        )
    [3] => Array
        (
            [name] => Samantha
            [age] => 50
            [hire_date] => 2000-12-14
        )
)

Here's why this is the correct output - the "name" field is sticky, so it cannot change its sort
order. Thus, the "age" field is only sorted as a sub-sort within records where "name" is
identical. Thus, the "Sam" records are reversed, because 44 > 23, but Samantha remains at the
bottom, even though her age is 50. This is a way of achieving "sub-sorts" and "sub-sub-sorts" (and
so on) within records of identical data for specific fields.

Courtesy of the $5 Script Archive: http://www.tufat.com
**/

define( 'ASC_AZ', 1000 );
define( 'DESC_AZ', 1001 );
define( 'ASC_NUM', 1002 );
define( 'DESC_NUM', 1003 );

function stickysort( $arr, $field, $sort_type, $sticky_fields = array() ) {
    $i = 0;
    foreach ($arr as $value) {
        $is_contiguous = true;
        if(!empty($grouped_arr)) {
            $last_value = end($grouped_arr[$i]);
```

```
            if(!($sticky_fields == array())) {
                foreach ($sticky_fields as $sticky_field) {
                    if ($value[$sticky_field] <> $last_value[$sticky_field]) {
                        $is_contiguous = false;
                        break;
                    }
                }
            }
        }
        if ($is_contiguous)
            $grouped_arr[$i][] = $value;
        else
            $grouped_arr[++$i][] = $value;
    }
    $code = '';
    switch($sort_type) {
        case ASC_AZ:
            $code .= 'return strcasecmp($a["'.$field.'"], $b["'.$field.'"]);';
            break;
        case DESC_AZ:
            $code .= 'return (-1*strcasecmp($a["'.$field.'"], $b["'.$field.'"]));';
            break;
        case ASC_NUM:
            $code .= 'return ($a["'.$field.'"] - $b["'.$field.'"]);';
            break;
        case DESC_NUM:
            $code .= 'return ($b["'.$field.'"] - $a["'.$field.'"]);';
            break;
    }

    $compare = create_function('$a, $b', $code);

    foreach($grouped_arr as $grouped_arr_key=>$grouped_arr_value)
        usort ( $grouped_arr[$grouped_arr_key], $compare );

    $arr = array();
    foreach($grouped_arr as $grouped_arr_key=>$grouped_arr_value)
        foreach($grouped_arr[$grouped_arr_key] as $grouped_arr_arr_key=>$grouped_arr_arr_value)
            $arr[] = $grouped_arr[$grouped_arr_key][$grouped_arr_arr_key];

    return $arr;
}
?>
```

up
down
0

Emiliyan at ServicesBG dot Com ¶

16 years ago

```
#This is a function that will sort an array...
function sort_by($array,  $keyname = null, $sortby) {
   $myarray = $inarray = array();
   # First store the keyvalues in a seperate array
    foreach ($array as $i => $befree) {
        $myarray[$i] = $array[$i][$keyname];
    }
   # Sort the new array by
    switch ($sortby) {
```

```
    case 'asc':
    # Sort an array and maintain index association...
    asort($myarray);
    break;
    case 'arsort':
    # Sort an array in reverse order and maintain index association
    arsort($myarray);
    break;
    case 'natcasesor':
    # Sort an array using a case insensitive "natural order" algorithm
    natcasesort($myarray);
    break;
    }
    # Rebuild the old array
    foreach ( $myarray as $key=> $befree) {
       $inarray[$key] = $array[$key];
    }
    return $inarray;
}
sort_by(); example...
$info = sort_by($myarray, 'name', $use = 'asc');
print_r($info);
```

up
down
0

nm at thenoodleman dot com ¶

16 years ago

Faster, more effective function:

```
array_sort (array, ['asc'/'desc'])

Second parameter specifies whether to order ascending or descending. Default is ascending.

function array_sort($array, $type='asc'){
    $result=array();
    foreach($array as $var => $val){
        $set=false;
        foreach($result as $var2 => $val2){
            if($set==false){
                if($val>$val2 && $type=='desc' || $val<$val2 && $type=='asc'){
                    $temp=array();
                    foreach($result as $var3 => $val3){
                        if($var3==$var2) $set=true;
                        if($set){
                            $temp[$var3]=$val3;
                            unset($result[$var3]);
                        }
                    }
                    $result[$var]=$val;
                    foreach($temp as $var3 => $val3){
                        $result[$var3]=$val3;
                    }
                }
            }
        }
        if(!$set){
            $result[$var]=$val;
        }
```

```
    }
    return $result;
}
```

Works for ordering by integers or strings, no need to specify which.

Example:

```
$array=array('a' => 50, 'b' => 25, 'c' => 75);
print_r(array_sort($array));
```

Returns:

```
Array
(
[b] => 25
[a] => 50
[c] => 75
)
```

up
down
0
timc at hlyw dot com ¶
17 years ago

I dig the multi_sort function(s) from above. But, they don't work for hash arrays. I added a keys variable to keep track of the key value as the array gets sorted. Feed back welcome.

```
<?php
function array_qsort (&$array, $column=0, $order=SORT_ASC, $first=0, $last= -2)
{
  // $array  - the array to be sorted
  // $column - index (column) on which to sort
  //          can be a string if using an associative array
  // $order  - SORT_ASC (default) for ascending or SORT_DESC for descending
  // $first  - start index (row) for partial array sort
  // $last  - stop  index (row) for partial array sort
  // $keys  - array of key values for hash array sort

  $keys = array_keys($array);
  if($last == -2) $last = count($array) - 1;
  if($last > $first) {
   $alpha = $first;
   $omega = $last;
   $key_alpha = $keys[$alpha];
   $key_omega = $keys[$omega];
   $guess = $array[$key_alpha][$column];
   while($omega >= $alpha) {
     if($order == SORT_ASC) {
       while($array[$key_alpha][$column] < $guess) {$alpha++; $key_alpha = $keys[$alpha]; }
       while($array[$key_omega][$column] > $guess) {$omega--; $key_omega = $keys[$omega]; }
     } else {
       while($array[$key_alpha][$column] > $guess) {$alpha++; $key_alpha = $keys[$alpha]; }
       while($array[$key_omega][$column] < $guess) {$omega--; $key_omega = $keys[$omega]; }
     }
     if($alpha > $omega) break;
     $temporary = $array[$key_alpha];
     $array[$key_alpha] = $array[$key_omega]; $alpha++;
     $key_alpha = $keys[$alpha];
     $array[$key_omega] = $temporary; $omega--;
```

```
      $key_omega = $keys[$omega];
    }
    array_qsort ($array, $column, $order, $first, $omega);
    array_qsort ($array, $column, $order, $alpha, $last);
   }
}
?>
```

up
down
-1

cmarshall at gmx dot de ¶
11 years ago

I read up on various problems re: sort() and German Umlaut chars and my head was soon spinning - bug in sort() or not, solution via locale or not, etc. ... (a total newbie here).

The obvious solution for me was quick and dirty: transform the Umlaut chars (present as HTML codes in my case) to their normal equivalent ('ä' = 'ae', 'ö' = 'oe', 'ü' = 'ue', 'ß' = 'ss' etc.), sort the array, then transform back. However there are cases in which a 'Mueller' is really that and does NOT need to be transformed into 'Müller' afterwards. Hence I for example replace the Umlaut itself with it's normal equivalent plus a char not used in the string otherwise (e.g. '_') so that the transfer back to Umlaut would only take place on certain combinations.

Of course any other char instead of '_' can be used as additional char (influencing the sort result). I know that my solution is rough at the edges and may cause other sort problems but it was sufficient for my purpose.

The array '$dat' in this example was filled with German town names (I actually worked with a multiple array ('$dat[][]') but stripped the code down to this as it's easier to understand):

```
<?php
// START Pre-sorting (Umlaut -> normal letters)
$max = count($dat);
for($totcnt = 0; $totcnt < $max; $totcnt++){
   $dat[$totcnt]=str_replace('&szlig;','ss_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&Auml;','Ae_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&auml;','ae_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&Ouml;','Oe_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&ouml;','oe_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&Uuml;','Ue_',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('&uuml;','ue_',$dat[$totcnt]);
}
// END Pre-sorting (Umlaut -> normal letters)

// START Sorting //
function compare_towns($a, $b)
{
return strnatcmp($a, $b);
}
usort($dat, 'compare_towns');
// END Sorting //

// START Post-sorting (normal letters -> Umlaut)
for($totcnt = 0; $totcnt < $max; $totcnt++){
   $dat[$totcnt]=str_replace('ss_','&szlig;',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('Ae_','&Auml;',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('ae_','&auml;',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('Oe_','&Ouml;',$dat[$totcnt]);
   $dat[$totcnt]=str_replace('oe_','&ouml;',$dat[$totcnt]);
```

```
    $dat[$totcnt]=str_replace('Ue_','&Uuml;',$dat[$totcnt]);
    $dat[$totcnt]=str_replace('ue_','&uuml;',$dat[$totcnt]);
}
// END Post-sorting (normal letters -> Umlaut)
?>
```

up
down
-1

poulou_0 at hotmail dot com ¶
11 years ago

if you are not interested in high or low case sort

```
<?php
//where
$sortable_array[$k] = $v2;
//put
$sortable_array[$k] = strtolower($v2);

//and where
$sortable_array[$k] = $v;
//put
$sortable_array[$k] = strtolower($v);
?>
```

up
down
-2

Brecht Cloetens ¶
12 years ago

```
<?php

/**
* function: array_columns
* author: Brecht Cloetens
* params: $a = array() // original array
*         $c = int() // number of columns
*/
function array_columns(&$a, $c=2)
{
    $m = ceil(count($a)/$c);
    $j = 0;
    for($i=0; $i<$m; $i++) {
        for($k=0; $k<$c; $k++) {
            $key = $i+($m*$k);
            settype($key,'integer');
            if(array_key_exists($key,$a)) {
                $b[$j] = $a[$key];
                $j++;
            }
        }
    }
    $a = $b;
}

$arr = range('a','z');
array_columns($arr,4);
print_r($arr);

?>
```

```
Example:
array(1,2,3,4,5) will be converted to array(1,4,2,5,3);

This can be easy if you want to display an array into a specified number of columns.

<table>
    <tr>
        <td>$arr[0] => 1</td>
        <td>$arr[1] => 4</td>
    </tr>
    <tr>
        <td>$arr[2] => 2</td>
        <td>$arr[3] => 5</td>
    </tr>
    <tr>
        <td>$arr[4] => 3</td>
        <td></td>
    </tr>
</table>
```

up
down
-2
raul at jimi dot com dot mx ¶
16 years ago

```
I had an array like this:
$arr=array (1,4,3,6,5);

which returns this:
$arr[0]=1
$arr[1]=4
$arr[2]=3
$arr[3]=6
$arr[4]=5

But lets say i remove [2] which is number 3, i get:

$arr[0]=1
$arr[1]=4
$arr[3]=6
$arr[4]=5

And i want to reindex without doing a sort because i dont want to lose the order of the numbers
(like a pop in a stack but in the middle of the list), i do this:

$arr=array_chunk($arr,count($arr));
$arr=$arr[0];

the result is:

$arr[0]=1
$arr[1]=4
$arr[2]=6
$arr[3]=5

This can be applied mostly for tree sorting, when you only have the id and the parent values of
the node, and you want to have N levels.
```

up

down
-3
sinan at sinaneldem dot com ¶
15 years ago

here is little script which will merge arrays, remove duplicates and sort it by alphabetical order:

```
<?php

$array1 = array('apple', 'banana','pear');
$array2 = array('grape', 'pear','orange');

function array_unique_merge_sort($array1, $array2){
$array = array_unique(array_merge($array1, $array2));
sort($array);
foreach ($array as $key => $value) {
    $new[$key] = $value;
}
return $new;
}

print_r (array_unique_merge_sort($array1, $array2));

?>
```

this will print out:

```
Array ( [0] => apple [1] => banana [2] => grape [3] => orange [4] => pear )
```

up
down
-3
anaz114119 at gmail dot com ¶
11 years ago

```
sort from textfile by coloumn
example name||date||time||comments
if you want to sort by date
$column = 2
<?php
function array_sort($array,$column){
$column = $column-1;
foreach($array as $line){
$bits = explode("||",$line);
$bits ="$bits[$column]**$line";
$array1[]=$bits;
}
asort($array1);
foreach($array1 as $line){
$bit = explode("**",$line);
$bit ="$bit[1]";
$array2[]=$bit;
}
return$array2;
}
?>
```

up
down
-3
anthony at ectrolinux dot com ¶

18 years ago

In a brief addition to the previous poster's message, the ascending sorting order used by PHP directly corresponds to ISO-8859-1 (ASCII). Therefore the character \48 (numeral 0) would be placed before the character \82 (R), which would be placed before the character \110 (n), and so forth.

up
down
-2

www at designdetector dot com ¶

14 years ago

To sort an array of multiple text fields alphabetically you have to make the text lowercase before sorting the array. Otherwise PHP puts acronyms before words. You can see this in my example code. Simply store the original text field at the end of the array line and call it later from there. You can safely ignore the lowercase version which is added to the start of the array line.

```
<?php
echo '<pre>ORIGINAL DATA:
<br />';

$data = array(
'Saturn|7|8|9|0||',
'Hello|0|1|2|3||',
'SFX|5|3|2|4||',
'HP|9|0|5|6||'
);

print_r($data);

sort($data);
reset($data);

echo '<br />RAW SORT:
<br />';

print_r($data);

for ($c = 0; $c < count($data); $c++) {
    list ($letter,$g1,$g2,$g3,$g4,$end) = explode ('|', $data[$c]);
    $lowercase = strtolower($letter);
    $data2[$c] = array($lowercase,$g1,$g2,$g3,$g4,$letter);
}

sort($data2);
reset($data2);

echo '<br />LOWERCASE SORT:
<br />';

print_r($data2);

echo '</pre>';
?>
```

up
down
-2

jesper at snt dot utwente dot nl ¶

16 years ago

If you sort an array of objects, the first variable in the object will be used for sorting:

```
<?php
class foo
{
  var $value; //First variable: Used for sorting
  var $id;

  function foo($i, $v)
  {
     $this->id = $i;
     $this->value = $v;
  }

}

for ($i = 0; $i < 10; $i++)
{
  $bar[] = new foo($i,rand(1,10));
}

// This will sort on value
sort($bar);
print_r($bar);
?>
```

Compare the piece of code above with the following:

```
<?php
class foo
{
  var $id; //First variable: Used for sorting
  var $value;

  function foo($i, $v)
  {
     $this->id = $i;
     $this->value = $v;
  }

}

for ($i = 0; $i = 10; $i++)
{
  $bar[] = new foo($i,rand(1,10));
}

// This will sort on id
sort($bar);
print_r($bar);
?>
```

As you can see the location of declaration of the variables matter!
If you want to sort on both or on a combination of variables, use ksort()

- Funciones de Arrays
  - array_change_key_case

- array_chunk
- array_column
- array_combine
- array_count_values
- array_diff_assoc
- array_diff_key
- array_diff_uassoc
- array_diff_ukey
- array_diff
- array_fill_keys
- array_fill
- array_filter
- array_flip
- array_intersect_assoc
- array_intersect_key
- array_intersect_uassoc
- array_intersect_ukey
- array_intersect
- array_is_list
- array_key_exists
- array_key_first
- array_key_last
- array_keys
- array_map
- array_merge_recursive
- array_merge
- array_multisort
- array_pad
- array_pop
- array_product
- array_push
- array_rand
- array_reduce
- array_replace_recursive
- array_replace
- array_reverse
- array_search
- array_shift
- array_slice
- array_splice
- array_sum
- array_udiff_assoc
- array_udiff_uassoc
- array_udiff
- array_uintersect_assoc
- array_uintersect_uassoc
- array_uintersect
- array_unique
- array_unshift
- array_values
- array_walk_recursive
- array_walk
- array
- arsort
- asort
- compact
- count
- current

- end
- extract
- in_array
- key_exists
- key
- krsort
- ksort
- list
- natcasesort
- natsort
- next
- pos
- prev
- range
- reset
- rsort
- shuffle
- sizeof
- sort
- uasort
- uksort
- usort

- Deprecated
  - each