Focus search box

next »
« natcasesort

- 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

natsort

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

natsort — Ordena un array usando un algoritmo de "orden natural"

Descripción ¶

natsort(array `&$array`): bool

Esta función implementa un algoritmo de ordenación que ordena las cadenas alfanuméricas en la manera en que lo haría un humano mientras mantiene las asociaciones de clave/valor. Es descrito como "ordenación natural". Un ejemplo de la diferencia entre este algoritmo y los algoritmos de ordenación normales de computadora (usados en sort()) se puede ver en el ejemplo de abajo.

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

Parámetros ¶

`array`

El array de entrada.

Valores devueltos ¶

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

Historial de cambios ¶

Versión	Descripción
5.2.10	Las cadenas numéricas rellenadas con ceros (p.ej., '00005') ahora ignoran el relleno de 0.

Ejemplos ¶

Ejemplo #1 Ejemplos de natsort() demostrando su uso básico

```
<?php
$array1 = $array2 = array("img12.png", "img10.png", "img2.png", "img1.png");
```

```php
asort($array1);
echo "Ordenación estándar\n";
print_r($array1);

natsort($array2);
echo "\nOrdenación de orden natural\n";
print_r($array2);
?>
```

El resultado del ejemplo sería:

```
Ordenación estándar
Array
(
    [3] => img1.png
    [1] => img10.png
    [0] => img12.png
    [2] => img2.png
)

Ordenación de orden natural
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Para más información véase: la página de Martin Pool » Natural Order String Comparison.

Ejemplo #2 Ejemplos de natsort() demostrando trampas potenciales

```php
<?php
echo "Números negativos\n";
$negativo = array('-5','3','-2','0','-1000','9','1');
print_r($negativo);
natsort($negativo);
print_r($negativo);

echo "Relleno de ceros\n";
$ceros = array('09', '8', '10', '009', '011', '0');
print_r($ceros);
natsort($ceros);
print_r($ceros);
?>
```

El resultado del ejemplo sería:

```
Números negativos
Array
(
    [0] => -5
    [1] => 3
    [2] => -2
    [3] => 0
    [4] => -1000
    [5] => 9
    [6] => 1
)
Array
(
    [2] => -2
```

```
    [0] => -5
    [4] => -1000
    [3] => 0
    [6] => 1
    [1] => 3
    [5] => 9
)

Relleno de ceros
Array
(
    [0] => 09
    [1] => 8
    [2] => 10
    [3] => 009
    [4] => 011
    [5] => 0
)
Array
(
    [5] => 0
    [1] => 8
    [3] => 009
    [0] => 09
    [2] => 10
    [4] => 011
)
```

Ver también ¶

- natcasesort() - Ordenar un array usando un algoritmo de "orden natural" insensible a mayúsculas-minúsculas
- comparación de funciones de orden de arrays
- strnatcmp() - Comparación de strings utilizando un algoritmo de "orden natural"
- strnatcasecmp() - Comparación de strings, insensible a mayúsculas y minúsculas, utilizando un algoritmo de "orden natural"

+ add a note

User Contributed Notes 19 notes

up
down
35
wyvern at greywyvern dot com ¶
13 years ago

There's no need to include your own API code to natsort an associative array by key. PHP's in-built functions (other than natsort) can do the job just fine:

```
<?php
  uksort($myArray, "strnatcmp");
?>
```

up
down
14
Johan GENNESSON (php at genjo dot fr) ¶
11 years ago

Be careful of the new behaviour in 5.2.10 version.
See the following sample:

```
<?php
```

```
$array = array('1 bis', '10 ter', '0 PHP', '0', '01', '01 Ver', '0 ', '1 ', '1');

natsort($array);
echo '<pre>';
print_r($array);
echo '</pre>';
?>

5.2.6-1 will output:
Array
(
    [3] => 0
    [6] => 0
    [2] => 0 OP
    [4] => 01
    [5] => 01 Ver
    [8] => 1
    [7] => 1
    [0] => 1 bis
    [1] => 10 ter
)

5.2.10 will output:
Array
(
    [6] => 0
    [3] => 0
    [8] => 1
    [4] => 01
    [7] => 1
    [5] => 01 Ver
    [0] => 1 bis
    [1] => 10 ter
    [2] => 0 OP
)

Greetings
```

up
down
10

flash at minet dot net ¶

19 years ago

```
About the reverse natsort.. Maybe simpler to do :

function strrnatcmp ($a, $b) {
    return strnatcmp ($b, $a);
}
```

up
down
9

xlab AT adaptiveNOSPAMarts DOT net ¶

18 years ago

```
Under limited testing, natsort() appears to work well for IP addresses. For my needs, it is far
less code than the ip2long()/long2ip() conversion I was using before.
```

up
down
5

rasmus at flajm dot com ¶
18 years ago
To make a reverse function, you can simply:

```
function rnatsort(&$a){
    natsort($a);
    $a = array_reverse($a, true);
}
```

up
down
4

mbirth at webwriters dot de ¶
18 years ago
For those who want to natsort a 2d-array on the first element of each sub-array, the following few lines should do the job.

```
<?php

function natsort2d(&$aryInput) {
  $aryTemp = $aryOut = array();
  foreach ($aryInput as $key=>$value) {
    reset($value);
    $aryTemp[$key]=current($value);
  }
  natsort($aryTemp);
  foreach ($aryTemp as $key=>$value) {
    $aryOut[] = $aryInput[$key];
  }
  $aryInput = $aryOut;
}

?>
```

up
down
4

anonymous at coward dot net ¶
19 years ago
Reverse Natsort:

```
  function rnatsort($a, $b) {
    return -1 * strnatcmp($a, $b);
  }

  usort($arr, "rnatsort");
```

up
down
3

natcasesort.too ¶
17 years ago
I got caught out through naive use of this feature - attempting to sort a list of image filenames from a digital camera, where the filenames are leading zero padded (e.g. DSCF0120.jpg) , will not sort correctly.
Maybe the example could be modified to exhibit this behaviour
(e.g. set array to -img0120.jpg','IMG0.png', 'img0012.png', 'img10.png', 'img2.png', 'img1.png', 'IMG3.png)
If the example hadn't used images I would have coded it correctly first time around!

up
down

2

ale152 ¶

13 years ago

Note: negatives number.

```
<?php
$a = array(-5,-2,3,9);
natsort($a);
print_r($a);
?>
```

Will output:

```
Array ( [1] => -2 [0] => -5 [2] => 3 [3] => 9 )
```

up

down

2

phpnet at moritz-abraham dot de ¶

18 years ago

additional to the code posted by justin at redwiredesign dot com (which I found very usefull) here is a function that sorts complex arrays like this:

```
<?
$array['test0'] = array('main' =>  'a', 'sub' => 'a');
$array['test2'] = array('main' =>  'a', 'sub' => 'b');
$array['test3'] = array('main' =>  'b', 'sub' => 'c');
$array['test1'] = array('main' =>  'a', 'sub' => 'c');
$array['test4'] = array('main' =>  'b', 'sub' => 'a');
$array['test5'] = array('main' =>  'b', 'sub' => 'b');
?>
```

or

```
<?
$array[0] = array('main' =>  1, 'sub' => 1);
$array[2] = array('main' =>  1, 'sub' => 2);
$array[3] = array('main' =>  2, 'sub' => 3);
$array[1] = array('main' =>  1, 'sub' => 3);
$array[4] = array('main' =>  2, 'sub' => 1);
$array[5] = array('main' =>  2, 'sub' => 2);
?>
```

on one or more columns.

the code

```
<? $array = array_natsort_list($array,'main','sub'); ?>
```

will result in $array being sortet like this:

test0,test2,test1,test4,test5,test3

or

0,2,1,4,5,3.

you may even submit more values to the function as it uses a variable parameter list. the function starts sorting on the last and the goes on until the first sorting column is reached.

to me it was very usefull for sorting a menu having submenus and even sub-submenus.

i hope it might help you too.

here is the function:

```
<?
function array_natsort_list($array) {
    // for all arguments without the first starting at end of list
    for ($i=func_num_args();$i>1;$i--) {
        // get column to sort by
        $sort_by = func_get_arg($i-1);
```

```
        // clear arrays
        $new_array = array();
        $temporary_array = array();
        // walk through original array
        foreach($array as $original_key => $original_value) {
            // and save only values
            $temporary_array[] = $original_value[$sort_by];
        }
        // sort array on values
        natsort($temporary_array);
        // delete double values
        $temporary_array = array_unique($temporary_array);
        // walk through temporary array
        foreach($temporary_array as $temporary_value) {
            // walk through original array
            foreach($array as $original_key => $original_value) {
                // and search for entries having the right value
                if($temporary_value == $original_value[$sort_by]) {
                    // save in new array
                    $new_array[$original_key] = $original_value;
                }
            }
        }
        // update original array
        $array = $new_array;
    }
    return $array;
}
?>
```

up
down
1

bb7b5b9 at gmail dot com ¶

6 years ago

This made me waste a lot of my precious youth ... natsort() is buggy if all numbers don't have the same number of decimal places.

(php 5.6.4-4ubuntu6.2)

```
<?php
$different_decimal_places_in_values = array('D'=>'13.59', '14.6' => '14.6', 'C-' => '14.19');
natsort($a);
var_dump($a);

/*echoes
array(3) {
  'D' =>
  string(5) "13.59"
  '14.6' =>
  string(4) "14.6" <----------- badly ordered
  'C-' =>
  string(5) "14.19"
}*/
?>
```

While this

```
<?php
```

```
$same_num_decimal_places_in_values = array('D'=>'13.59', '14.6' => '14.60', 'C-' => '14.19');
natsort($a); var_dump($a);

/*echoes
array(3) {
  'D' =>
  string(5) "13.59"
  'C-' =>
  string(5) "14.19"
  '14.6' =>
  string(5) "14.60" <--------- that is the correct position
}
*/

?>
```

up
down
1

lacent at gmail dot com ¶

15 years ago

there is another rnatsort function lower on the page, but it didn't work in the context i needed it in.

reasoning for this:
sorting naturally via the keys of an array, but needing to reverse the order.

```
    function rnatsort ( &$array = array() )
    {
        $keys    = array_keys($array);
        natsort($keys);
        $total    = count($keys) - 1;
        $temp1    = array();
        $temp2     = array();

        // assigning original keys to an array with a backwards set of keys, to use in krsort();
        foreach ( $keys as $key )
        {
            $temp1[$total] = $key;
            --$total;
        }

        ksort($temp1);

        // setting the new array, with the order from the krsort() and the values of original
array.
        foreach ( $temp1 as $key )
        {
            $temp2[$key] = $array[$key];
        }

        $array = $temp2;
    }
```

up
down
1

h3 ¶

16 years ago

This function can be very usefull, but in some cases, like if you want to sort a MySQL query result, it's important to keep in mind that MySQL as built'in sorting functions which are way faster than resorting the result using a complex php algorythm, especially with large arrays.

ex; 'SELECT * FROM `table` ORDER BY columnName ASC, columnName2 DESC'

up
down
1

Malek Mohamed ¶

4 years ago

```
$array1 = $array2 = array('IMG0.png', 'img12.png', 'img10.png', 'img2.png', 'img1.png',
'IMG3.png');

natsort($array1);
echo "\n natsort(); \n";
print_r($array1);

sort($array2, SORT_NATURAL);
echo "\n sort() with  SORT_NATURAL Option\n";
print_r($array2);
```

Ouput:

```
natsort();
Array
(
    [0] => IMG0.png
    [5] => IMG3.png
    [4] => img1.png
    [3] => img2.png
    [2] => img10.png
    [1] => img12.png
)

sort() with  SORT_NATURAL Option
Array
(
    [0] => IMG0.png
    [1] => IMG3.png
    [2] => img1.png
    [3] => img2.png
    [4] => img10.png
    [5] => img12.png
)
```

as we can see it's the same values but not the same keys, and also it's same for sort($array1, SORT_NATURAL | SORT_FLAG_CASE); and natcasesort($array2)

up
down
0

mvs dot php at gmail dot com ¶

7 years ago

To naturally sort by array key, the uksort function can be used.

```
<?php

echo "Sort by keys\n";
$smoothie = array('orange' => 1, 'apple' => 1, 'yogurt' => 4, 'banana' => 4);
```

```
print_r($smoothie);
uksort( $smoothie, 'strnatcmp');
print_r($smoothie)

?>
```

Output:

```
Sort by keys
Array
(
    [orange] => 1
    [apple]  => 1
    [yogurt] => 4
    [banana] => 4
)
Array
(
    [apple]  => 1
    [banana] => 4
    [orange] => 1
    [yogurt] => 4
)
```

See http://php.net/manual/en/function.uksort.php for more information about uksort and http://php.net/strnatcmp for usage of strnatcmp.

up
down
0

@gmail bereikme ¶

16 years ago

Here's a handy function to sort an array on 1 or more columns using natural sort:

```
<?php
// Example: $records = columnSort($records, array('name', 'asc', 'addres', 'desc', 'city',
'asc'));

$globalMultisortVar = array();
function columnSort($recs, $cols) {
    global $globalMultisortVar;
    $globalMultisortVar = $cols;
    usort($recs, 'multiStrnatcmp');
    return($recs);
}

function multiStrnatcmp($a, $b) {
    global $globalMultisortVar;
    $cols = $globalMultisortVar;
    $i = 0;
    $result = 0;
    while ($result == 0 && $i < count($cols)) {
        $result = ($cols[$i + 1] == 'desc' ? strnatcmp($b[$cols[$i]], $a[$cols[$i]]) : $result =
strnatcmp($a[$cols[$i]], $b[$cols[$i]]));
        $i+=2;
    }
    return $result;
}

?>
```

```
Greetings,

  - John
```

up
down
-2
AJenbo ¶
13 years ago

natsort might not act like you would expect with zero padding, heres a quick sample.

```
<?php
$array = array('09', '8', '10', '009', '011');
natsort($array);
?>
/*
Array
(
    [3] => 009
    [4] => 011
    [0] => 09
    [1] => 8
    [2] => 10
)
*/
```

up
down
-1
lil at thedreamersmaze dot spam-me-not dot org ¶
16 years ago

There's one little thing missing in this useful bit of code posted by mbirth at webwriters dot de:

```
<?php

function natsort2d(&$aryInput) {
  $aryTemp = $aryOut = array();
  foreach ($aryInput as $key=>$value) {
   reset($value);
   $aryTemp[$key]=current($value);
  }
  natsort($aryTemp);
  foreach ($aryTemp as $key=>$value) {
   $aryOut[$key] = $aryInput[$key];
// --------^^^^ add this if you want your keys preserved!
  }
  $aryInput = $aryOut;
}

?>
```

up
down
-1
dotancohen splat gmail spot com ¶
5 years ago

As noted in other notes, natsort() does _not_ always return the expected sort order. It seems especially buggy when decimals or 0 padding is used. I've filed this bug report on the issue: https://bugs.php.net/bug.php?id=74672

+ add a note

- 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

- My PHP.net
- Contact
- Other PHP.net sites
- Privacy policy
- View Source