Focus search box

array_push »
« array_pop

- 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

array_product

(PHP 5 >= 5.1.0, PHP 7, PHP 8)

array_product — Calcula el producto de los valores de un array

Descripción ¶

array_product(array `$array`): number

array_product() devuelve el producto de valores de un array.

Parámetros ¶

`array`

El array.

Valores devueltos ¶

Devuelve el producto como integer o float.

Historial de cambios ¶

Versión	Descripción
5.3.6	El producto de un array vacío ahora es 1, mientras que antes esta función devolvía 0 para un array vacío.

Ejemplos ¶

Ejemplo #1 Ejemplo de array_product()

```php
<?php

$a = array(2, 4, 6, 8);
echo "producto(a) = " . array_product($a) . "\n";
echo "producto(array()) = " . array_product(array()) . "\n";

?>
```

El resultado del ejemplo sería:

```
producto(a) = 384
producto(array()) = 1
```

+ add a note

User Contributed Notes 6 notes

up
down
29
***Andre D* ¶**
16 years ago

```
This function can be used to test if all values in an array of booleans are TRUE.

Consider:

<?php

function outbool($test)
{
    return (bool) $test;
}

$check[] = outbool(TRUE);
$check[] = outbool(1);
$check[] = outbool(FALSE);
$check[] = outbool(0);

$result = (bool) array_product($check);
// $result is set to FALSE because only two of the four values evaluated to TRUE

?>

The above is equivalent to:

<?php

$check1 = outbool(TRUE);
$check2 = outbool(1);
$check3 = outbool(FALSE);
$check4 = outbool(0);

$result = ($check1 && $check2 && $check3 && $check4);

?>

This use of array_product is especially useful when testing an indefinite number of booleans and
is easy to construct in a loop.
```

up
down
10
***bsr dot anwar at gmail dot com* ¶**
5 years ago

```
Here's how you can find a factorial of a any given number with help of range and array_product
functions.

function factorial($num) {
    return array_product(range(1, $num));
}

printf("%d", factorial(5)); //120
```

up
down
0
biziclop ¶
17 days ago

```
You can use array_product() to calculate the geometric mean of an array of numbers:

<?php
$a = [ 1, 10, 100 ];
$geom_avg = pow( array_product( $a ), 1 / count( $a ));
// = 9.999999999999998 ≈ 10
?>
```

up
down
-1
Marcel G ¶
12 years ago

```
You can use array_product to calculate the factorial of n:
<?php
function factorial( $n )
{
  if( $n < 1 ) $n = 1;
  return array_product( range( 1, $n ));
}
?>

If you need the factorial without having array_product available, here is one:
<?php
function factorial( $n )
{
  if( $n < 1 ) $n = 1;
  for( $p++; $n; ) $p *= $n--;
  return $p;
}
?>
```

up
down
-2
Jimmy PHP ¶
8 years ago

```
array_product() can be used to implement a simple boolean AND search

<?php
$args = array('first_name'=>'Bill','last_name'=>'Buzzard');
$values[] = array('first_name'=>'Brenda','last_name'=>'Buzzard');
$values[] = array('first_name'=>'Victor','last_name'=>'Vulture');
$values[] = array('first_name'=>'Bill','last_name'=>'Blue Jay');
$values[] = array('first_name'=>'Bill','last_name'=>'Buzzard');

$result = search_for($values,$args);
var_dump($result);exit;

function search_for($array,$args) {
    $results = array();
    foreach ($array as $row) {
        $found = false;
        $hits = array();
        foreach ($row as $k => $v) {
            if (array_key_exists($k,$args)) $hits[$k] = ($args[$k] == $v);
        }

        $found = array_product($hits);
        if (!in_array($row,$results) && true == $found) $results[] = $row;
    }

    return $results;
}
?>

Output:

array (size=1)
  0 =>
    array (size=2)
      'first_name' => string 'Bill' (length=4)
      'last_name' => string 'Buzzard' (length=7)
```

up
down
-7
pqpqpq at wanadoo dot nl ¶
15 years ago

```
An observation about the _use_ of array_product with primes:

$a=$arrayOfSomePrimes=(2,3,11);
              // 2 being the first prime (these days)

$codeNum=array_product($a); // gives 66 (== 2*3*11)

echo "unique product(\$a) = " . array_product($a) . "\n";

The 66 can (only) be split into its original primes,
which can be transformed into their place in the row of primes (2,3,5,7,11,13,17,19...)  giving
(1,2,3,4,5,6,7,8...)

The 66 gives the places {1,2,5} in the row of primes. The number "66" is unique as a code for
{1,2,5}

So you can define the combination of table-columns {1,2,5} in "66". The bigger the combination,
the more efficient in memory/transmission, the less in calculation.
```

+ 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