ord »
« nl2br

- Manual de PHP
- Referencia de funciones
- Procesamiento de texto
- Strings
- Funciones de strings

Change language: Spanish

Submit a Pull Request Report a Bug

number_format

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

number_format — Formatear un número con los millares agrupados

Descripción ¶

number_format(float `$number`, int `$decimals` = 0): string
number_format(
    float `$number`,
    int `$decimals` = 0,
    string `$dec_point` = ".",
    string `$thousands_sep` = ","
): string

Esta función acepta uno, dos, o cuatro parámetros (no tres):

Si solo se proporciona un parámetro, `number` será formateado sin decimales, pero con una coma (",") entre cada grupo de millares.

Si se proporcionan dos parámetros, `number` será formateado con tantos decimales como se hayan definido en `decimals` colocando un punto (".") delante, y una coma (",") entre cada grupo de millares.

Si se proporcionan los cuatro parámetros, `number` será formateado con tantos decimales como hayamos definido en `decimals`, `dec_point` sustituirá al punto (".") como seperador de los decimales y el separador por defecto de los millares, la coma (","), será sustituida por `thousands_sep`.

Parámetros ¶

`number`

El número al que dar formato.

`decimals`

Establece el número de puntos decimales.

`dec_point`

Establece el separador para los decimales.

`thousands_sep`

Establece el separador para los millares.

Valores devueltos ¶

Una versión formateada de `number`.

Historial de cambios ¶

Versión	Descripción
5.4.0	Esta función ahora admite múltiples bytes en `dec_point` y `thousands_sep`. En versiones anteriors solamente se utilizaba el primer byte de cada separador.

Ejemplos ¶

Ejemplo #1 Ejemplo de number_format()

Por ejemplo, la notación francesa suele utilizar dos decimales, una coma (',') como separador decimal y el espacio (' ') como separador de millares. Esto consigue esto mediante las siguientes líneas:

```
<?php

$número = 1234.56;

// notación inglesa (por defecto)
$número_formato_inglés = number_format($número);
// 1,235

// notación francesa
$nombre_format_francais = number_format($número, 2, ',', ' ');
// 1 234,56

$número = 1234.5678;

// notación inglesa sin separador de millares
$english_format_number = number_format($número, 2, '.', '');
// 1234.57

?>
```

Ver también ¶

- money_format() - Da formato a un número como un string de moneda
- sprintf() - Devuelve un string formateado
- printf() - Imprimir una cadena con formato
- sscanf() - Interpreta un string de entrada de acuerdo con un formato

+ add a note

User Contributed Notes 36 notes

up
down
370
thomas at weblizards dot de ¶
13 years ago
It's not explicitly documented; number_format also rounds:

```
<?php
```

```
$numbers = array(0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009);
foreach ($numbers as $number)
    print $number."->".number_format($number, 2, '.', ',')."<br>";
?>

0.001->0.00
0.002->0.00
0.003->0.00
0.004->0.00
0.005->0.01
0.006->0.01
0.007->0.01
0.008->0.01
0.009->0.01
```

up
down
8
info at ensostudio dot ru ¶
10 months ago

```
Note: use NumberFormatter to convert in human-readable format instead  user function from
comments:
<?php
echo NumberFormatter::create('en', NumberFormatter::SPELLOUT)->format(12309); // twelve thousand
three hundred nine
echo NumberFormatter::create('ru', NumberFormatter::SPELLOUT)->format(12307.5); //  двенадцать
тысяч триста семь целых пять десятых
?>
```

up
down
39
james at bandit dot co.nz ¶
13 years ago

```
Outputs a human readable number.

<?php
    #    Output easy-to-read numbers
    #    by james at bandit.co.nz
    function bd_nice_number($n) {
        // first strip any formatting;
        $n = (0+str_replace(",","",$n));

        // is this a number?
        if(!is_numeric($n)) return false;

        // now filter it;
        if($n>1000000000000) return round(($n/1000000000000),1).' trillion';
        else if($n>1000000000) return round(($n/1000000000),1).' billion';
        else if($n>1000000) return round(($n/1000000),1).' million';
        else if($n>1000) return round(($n/1000),1).' thousand';

        return number_format($n);
    }
?>

Outputs:

247,704,360 -> 247.7 million
866,965,260,000 -> 867 billion
```

up
down
24
MarcM ¶
16 years ago
For Zero fill - just use the sprintf() function

```
$pr_id = 1;
$pr_id = sprintf("%03d", $pr_id);
echo $pr_id;

//outputs 001
-----------------

$pr_id = 10;
$pr_id = sprintf("%03d", $pr_id);
echo $pr_id;

//outputs 010
-----------------
```

You can change %03d to %04d, etc.
up
down
18
stm555 at hotmail dot com ¶
17 years ago
I ran across an issue where I wanted to keep the entered precision of a real value, without arbitrarily rounding off what the user had submitted.

I figured it out with a quick explode on the number before formatting. I could then format either side of the decimal.

```
<?php
      function number_format_unlimited_precision($number,$decimal = '.')
      {
           $broken_number = explode($decimal,$number);
           return number_format($broken_number[0]).$decimal.$broken_number[1];
      }
?>
```

up
down
4
Lio ¶
4 years ago
Be carreful, when you're using French notation
means : number_format(124.25, 2 , ',' , ' ') with ',' as dec_point,

Don't forget to specify thousands_sep that default is ',' to another value, otherwise function will return null.
up
down
4
nospam at nospam dot com ¶
13 years ago
Simple function to show money as only dollars if no cents, but will show 2 decimals if cents exist.

```
The 'cents' flag can force to never or always show 2 decimals

<?php

// formats money to a whole number or with 2 decimals; includes a dollar sign in front
function formatMoney($number, $cents = 1) { // cents: 0=never, 1=if needed, 2=always
  if (is_numeric($number)) { // a number
    if (!$number) { // zero
      $money = ($cents == 2 ? '0.00' : '0'); // output zero
    } else { // value
      if (floor($number) == $number) { // whole number
        $money = number_format($number, ($cents == 2 ? 2 : 0)); // format
      } else { // cents
        $money = number_format(round($number, 2), ($cents == 0 ? 0 : 2)); // format
      } // integer or decimal
    } // value
    return '$'.$money;
  } // numeric
} // formatMoney

$a = array(1, 1234, 1.5, 1.234, 2.345, 2.001, 2.100, '1.000', '1.2345', '12345', 0, '0.00');

// show cents if needed ($cents=1)
foreach ($a as $b) echo ('<br />'.$b.' = '.formatMoney($b, 1));
1 = $1
1234 = $1,234
1.5 = $1.50
1.234 = $1.23
2.345 = $2.35
2.001 = $2.00
2.1 = $2.10
1.000 = $1
1.2345 = $1.23
12345 = $12,345
0 = $0
0.00 = $0

// never show cents ($cents=0)
foreach ($a as $b) echo ('<br />'.$b.' = '.formatMoney($b, 0));
1 = $1
1234 = $1,234
1.5 = $2
1.234 = $1
2.345 = $2
2.001 = $2
2.1 = $2
1.000 = $1
1.2345 = $1
12345 = $12,345
0 = $0
0.00 = $0

// always show cents ($cents=2)
foreach ($a as $b) echo ('<br />'.$b.' = '.formatMoney($b, 2));
1 = $1.00
1234 = $1,234.00
1.5 = $1.50
1.234 = $1.23
```

```
2.345 = $2.35
2.001 = $2.00
2.1 = $2.10
1.000 = $1.00
1.2345 = $1.23
12345 = $12,345.00
0 = $0.00
0.00 = $0.00

?>
```

Cheers :)

And remember to always contribute custom functions if they might be useful to the rest of us or future versions of the php language.

up
down
10

sgj at dr dot com ¶

19 years ago

Just an observation:
The number_format rounds the value of the variable.

```
$val1 = 1.233;
$val2 = 1.235;
$val3 = 1.237;

echo number_format($val1,2,",","."); // returns: 1,23
echo number_format($val2,2,",","."); // returns: 1,24
echo number_format($val3,2,",","."); // returns: 1,24
```

up
down
2

tomislav at firmus-grupa dot hr ¶

9 years ago

When apply number_format on number with separator on thousands, result is wrong. This function accept number of any format

```
<?php
function format_price($number,$decPlaces,$decSep,$thouSep){

    //$number - number for format
    //$decPlaces - number of decimal places
    //$decSep - separator for decimals
    //$thouSep - separator for thousands

    //first remove all white spaces
    $number=preg_replace('/\s+/', '',$number);
    //split string into array
    $numberArr = str_split($number);
    //reverse array and not preserve key, keys will help to find decimal place
    $numberArrRev=array_reverse($numberArr);

    //find first occurrence of non number character, that will be a decimal place
    //store $key into variable $decPointIsHere
    foreach ($numberArrRev as $key => $value) {
        if(!is_numeric($value)){
            if($decPointIsHere==""){
                $decPointIsHere=$key;
```

```
            }
        }
    }

    //decimal comma or whatever it is replace with dot
    //$decPointIsHere is the key of the element that will contain decimal separator dot
    if($decPointIsHere!=""){
        $numberArrRev[$decPointIsHere]=".";
    }

    //again check through array for non numerical characters but skipping allready processed keys
    //if is not number remove from array

    foreach ($numberArrRev as $key => $value) {
        if(!is_numeric($value) && $key>$decPointIsHere){
            unset($numberArrRev[$key]);
        }
    }

    //reverse back, at the start reversed array $numberArrRev to $numberArr
    $numberArr=array_reverse($numberArrRev);

    //create string from array
    $numberClean=implode("",$numberArr);

    // apply php number_format function
    return number_format($numberClean,$decPlaces,$decSep,$thouSep);

}

echo format_price("1 225 552, 55",2,',',' ')."<br>";
echo format_price("1.225.552, 55",2,',',' ')."<br>";
echo format_price("1'225'552. 55",2,',',' ')."<br>";
echo format_price("1225552.55",2,',',' ')."<br>";
?>
all results are: 1 225 552,55
```

up
down
6
isapoetra at gmail dot com ¶
14 years ago

here is the code to convert number to Indonesian text, this code has limitation as is number_format function. sorry for this.

```
/*
* Created : Iwan Sapoetra - Jun 13, 2008
* Project : Web
* Package : cgaf
*
*/
function terbilang( $num ,$dec=4){
    $stext = array(
        "Nol",
        "Satu",
        "Dua",
        "Tiga",
        "Empat",
        "Lima",
        "Enam",
```

```
        "Tujuh",
        "Delapan",
        "Sembilan",
        "Sepuluh",
        "Sebelas"
    );
    $say  = array(
        "Ribu",
        "Juta",
        "Milyar",
        "Triliun",
        "Biliun", // remember limitation of float
        "--apaan---" ///setelah biliun namanya apa?
    );
    $w = "";

    if ($num <0 ) {
        $w  = "Minus ";
        //make positive
        $num *= -1;
    }

    $snum = number_format($num,$dec,",",".");
    die($snum);
    $strnum =  explode(".",substr($snum,0,strrpos($snum,",")));
    //parse decimalnya
    $koma = substr($snum,strrpos($snum,",")+1);

    $isone = substr($num,0,1)  ==1;
    if (count($strnum)==1) {
        $num = $strnum[0];
        switch (strlen($num)) {
            case 1:
            case 2:
                if (!isset($stext[$strnum[0]])){
                    if($num<19){
                        $w .=$stext[substr($num,1)]." Belas";
                    }else{
                        $w .= $stext[substr($num,0,1)]." Puluh ".
                            (intval(substr($num,1))==0 ? "" : $stext[substr($num,1)]);
                    }
                }else{
                    $w .= $stext[$strnum[0]];
                }
                break;
            case 3:
                $w .=  ($isone ? "Seratus" : terbilang(substr($num,0,1)) .
                    " Ratus").
                    " ".(intval(substr($num,1))==0 ? "" : terbilang(substr($num,1)));
                break;
            case 4:
                $w .=  ($isone ? "Seribu" : terbilang(substr($num,0,1)) .
                    " Ribu").
                    " ".(intval(substr($num,1))==0 ? "" : terbilang(substr($num,1)));
                break;
            default:
                break;
        }
```

```
    }else{
        $text = $say[count($strnum)-2];
        $w = ($isone && strlen($strnum[0])==1 && count($strnum) <=3? "Se".strtolower($text) :
terbilang($strnum[0]).' '.$text);
        array_shift($strnum);
        $i =count($strnum)-2;
        foreach ($strnum as $k=>$v) {
            if (intval($v)) {
                $w.= ' '.terbilang($v).' '.($i >=0 ? $say[$i] : "");
            }
            $i--;
        }
    }
    $w = trim($w);
    if ($dec = intval($koma)) {
        $w .= " Koma ". terbilang($koma);
    }
    return trim($w);
}
//example
echo terbilang(999999999999)."\n";
/**
* result : Sembilan Ratus Sembilan Puluh Sembilan Milyar Sembilan Ratus Sembilan Puluh Sembilan
Juta Sembilan Ratus Sembilan Puluh Sembilan Ribu Sembilan Ratus Sembilan Puluh Sembilan
*/
echo terbilang(9999999999999999);
/**
* todo : fix this bug pleasese
* problem : number_format(9999999999999999) <--- 10.000.000.000.000.000,0000
* Result : Sepuluh Biliun
*/
```

up
down
1

keyg at auralplanet dot com ¶
17 years ago

if you want   as a separator and use windows charset this piece of code may help:

```
<?php
$number=number_format($number,2,'.',chr(0xA0));
?>
```

up
down
3

Theo Diem ¶
19 years ago

formatting numbers may be more easy if u use number_format function.

I also wrote this :

```
function something($number)
{
    $locale = localeconv();
    return number_format($number,
       $locale['frac_digits'],
        $locale['decimal_point'],
        $locale['thousands_sep']);
}
```

hope this helps =)
[]'s
up
down
2
liviu andrei (bls) ¶
10 years ago
To prevent the rounding that occurs when next digit after last significant decimal is 5 (mentioned by several people below):

```
<?php
function fnumber_format($number, $decimals='', $sep1='', $sep2='') {

        if (($number * pow(10 , $decimals + 1) % 10 ) == 5)  //if next not significant digit is 5
            $number -= pow(10 , -($decimals+1));

        return number_format($number, $decimals, $sep1, $sep2);

}

$t=7.15;
echo $t . " | " . number_format($t, 1, '.', ',') .  " | " . fnumber_format($t, 1, '.', ',') .
"\n\n";
//result is: 7.15 | 7.2 | 7.1

$t=7.3215;
echo $t . " | " . number_format($t, 3, '.', ',') .  " | " . fnumber_format($t, 3, '.', ',') .
"\n\n";
//result is: 7.3215 | 7.322 | 7.321
} ?>
```

have fun!
up
down
2
xmontero at dsitelecom dot com ¶
7 months ago
You can use:

```
    number_format( $number, '.', ' ' )
```

Rationale:

Since 2003, the Internationa Standard of Units regulated that groups of three are to be separated by spaces and not dots or commas:

We can read about the S.I. in wikipedia about the numbers:

"Spaces should be used as a thousands separator (1000000) in contrast to commas or periods (1,000,000 or 1.000.000) to reduce confusion resulting from the variation between these forms in different countries."

here https://en.wikipedia.org/wiki/International_System_of_Units in the section "General Rules".

More specifically in an official document they introduce the concept of "thin space":

"for numbers with many digits the digits may be divided into groups of three by a thin space, in order to facilitate reading. Neither dots nor commas are inserted in the spaces between groups of

three."

here https://mcyt.educa.madrid.org/laboratorios/Documentos/Otros/documentos/si_brochure_8_en.pdf in page 41, in the section "5.3.4 Formatting numbers, and the decimal marker".

Finally, also in wikipedia, we can read about the separation of digits:

"It is also used in the International System of Units and in many countries as a thousands separator when writing numbers in groups of three digits, in order to facilitate reading."

here: https://en.wikipedia.org/wiki/Thin_space

Note that "THIN SPACE" is breakable and for a non-breakable space we also have the "NARROW NO-BREAK SPACE". Nevertheless the definition is a bit different:

"Also starting from release 34 of Unicode Common Locale Data Repository (CLDR) the NNBSP is used in numbers as thousands group separator for French and Spanish locale."

We can read it here: https://en.wikipedia.org/wiki/Non-breaking_space at the end of the article

So, conclusion: We can use   for the regular one or   for the non-breaking one in the number_format function to accomodate modern times and forget once and for all to use dots or commas for the thousands.

More literature:

* Archive of the PDF https://web.archive.org/web/20140705194729/http://www.bipm.org/en/si/si_brochure/chapter5/5-3-2.html

* https://www.bipm.org/en/committees/cg/cgpm/22-2003/resolution-10

up
down
1

zulisse at email dot it ¶

15 years ago

simpler function to convert a number in bytes, kilobytes....

```
<?php

function bytes($a) {
    $unim = array("B","KB","MB","GB","TB","PB");
    $c = 0;
    while ($a>=1024) {
        $c++;
        $a = $a/1024;
    }
    return number_format($a,($c ? 2 : 0),",",".")." ".$unim[$c];
}

?>
```

you may also add others units over PeraBytes when the hard disks will reach 1024 PB :)

up
down
0

Anonymous ¶

2 months ago

Note: Changing the number format loses PHP's ability to count. So do not change number format if you wish to do anything besides showing the number.

up
down
1

besciualexandru at gmail dot com ¶

7 years ago

```
// Here is a function that produces the same output as number_format() but also works with numbers bigger than 2^53.

function a_number_format($number_in_iso_format, $no_of_decimals=3, $decimals_separator='.', $thousands_separator='', $digits_grouping=3){
    // Check input variables
    if (!is_numeric($number_in_iso_format)){
        error_log("Warning! Wrong parameter type supplied in my_number_format() function. Parameter \$number_in_iso_format is not a number.");
        return false;
    }
    if (!is_numeric($no_of_decimals)){
        error_log("Warning! Wrong parameter type supplied in my_number_format() function. Parameter \$no_of_decimals is not a number.");
        return false;
    }
    if (!is_numeric($digits_grouping)){
        error_log("Warning! Wrong parameter type supplied in my_number_format() function. Parameter \$digits_grouping is not a number.");
        return false;
    }


    // Prepare variables
    $no_of_decimals = $no_of_decimals * 1;


    // Explode the string received after DOT sign (this is the ISO separator of decimals)
    $aux = explode(".", $number_in_iso_format);
    // Extract decimal and integer parts
    $integer_part = $aux[0];
    $decimal_part = isset($aux[1]) ? $aux[1] : '';

    // Adjust decimal part (increase it, or minimize it)
    if ($no_of_decimals > 0){
        // Check actual size of decimal_part
        // If its length is smaller than number of decimals, add trailing zeros, otherwise round it
        if (strlen($decimal_part) < $no_of_decimals){
            $decimal_part = str_pad($decimal_part, $no_of_decimals, "0");
        } else {
            $decimal_part = substr($decimal_part, 0, $no_of_decimals);
        }
    } else {
        // Completely eliminate the decimals, if there $no_of_decimals is a negative number
        $decimals_separator = '';
        $decimal_part       = '';
    }

    // Format the integer part (digits grouping)
    if ($digits_grouping > 0){
```

```
        $aux = strrev($integer_part);
        $integer_part = '';
        for ($i=strlen($aux)-1; $i >= 0 ; $i--){
            if ( $i % $digits_grouping == 0 && $i != 0){
                $integer_part .= "{$aux[$i]}{$thousands_separator}";
            } else {
                $integer_part .= $aux[$i];
            }
        }
    }

    $processed_number = "{$integer_part}{$decimals_separator}{$decimal_part}";
    return $processed_number;
}

$original_number= 9223372036854775805;
echo a_number_format($original_number, 4, '.',"'",3);
// Outputs: 9'223'372'036'854'775'805.1230
```

up
down
0

oelschlegel at gmail dot com ¶

9 months ago

Setting the second argument to a value to greater than what is possible in floating point representation returns some interesting results.

```
<?php

$a = 1234.5678;
print(number_format($a, 14, '.', ''));

// 1234.56780000000003

?>
```

up
down
0

info at ensostudio dot ru ¶

10 months ago

Auto calculate decimals by default:

```
<?php
function numberFormat(
    float $num,
    int $decimals = -1,
    ?string $decimalSeparator = '.',
    ?string $thousandsSeparator = ''
): string {
    if ($decimals < 0) {
        $intNum = (int) $num;
        if ($num == $intNum) {
            $decimals = 0;
        } else {
            $decimals = strlen($num) - strlen($intNum) - 1;
        }
    }
    return number_format($num, $decimals, $decimalSeparator, $thousandsSeparator);
}
?>
```

up
down
2
info at daniel-marschall dot de ¶
13 years ago

In my function my_number_format() [shown below] there was a bug.

If a negative number which is smaller than 1 was entered (-0,...), then the result was wrongly positive because +0 is equal to -0 (the content of $tmp[0] which was interpretet as numeric value).

Here is the corrected version:

```
<?php

function my_number_format($number, $dec_point, $thousands_sep)
{
    $was_neg = $number < 0; // Because +0 == -0
    $number = abs($number);

    $tmp = explode('.', $number);
    $out = number_format($tmp[0], 0, $dec_point, $thousands_sep);
    if (isset($tmp[1])) $out .= $dec_point.$tmp[1];

    if ($was_neg) $out = "-$out";

    return $out;
}

?>
```

Thanks to Federico Cassinelli for the bug report.

[EDIT BY danbrown AT php DOT net: The original note follows.]

Let's say we got the number $inp = 1234.56

By using

```
<?php
return number_format($inp, 2, ',', '.');
?>
```

you can get the German format 1.234,56. (Comma as decimal separator and point as thousand separator)

But I have a problem with that: I want to add commas as thousand separators and change the decimal-separator (this could also be done with str_replace), but I do not want to change the amount of fractional digits!

But since the 2nd argument of number_format is necessary to enter the 3rd and 4th argument, this cannot be done with number_format. You have to change the fractional digits with this function.

But I want that 1234.56 changes into 1.234,56 and 1234.567890123456 changes into 1.234,567890123456

So, I created following function, that doesn't change the amount of fractional digits:

```
<?php
function my_number_format($number, $dec_point, $thousands_sep)
{
  $tmp = explode('.', $number);
  $out = number_format($tmp[0], 0, $dec_point, $thousands_sep);
  if (isset($tmp[1])) $out .= $dec_point.$tmp[1];

  return $out;
}
?>
```

up
down
0

markagius at markagius co uk ¶

2 years ago

If you want a number of digits after the point, but not unnecessary zeros.
Eg.
number_format(1.20000,4) = 1.2000
num_format(1.20000,4,0) = 1.2

number_format(1.20000,4) = 1.2000
num_format(1.20000,4,2) = 1.20

number_format(1.23456,4) = 1.2345
num_format(1.23456,4,2) = 1.2345

```
function num_format($numVal,$afterPoint=2,$minAfterPoint=0,$thousandSep=",",$decPoint="."){
  // Same as number_format() but without unnecessary zeros.
  $ret = number_format($numVal,$afterPoint,$decPoint,$thousandSep);
  if($afterPoint!=$minAfterPoint){
    while(($afterPoint>$minAfterPoint) && (substr($ret,-1) =="0") ){
      // $minAfterPoint!=$minAfterPoint and number ends with a '0'
      // Remove '0' from end of string and set $afterPoint=$afterPoint-1
      $ret = substr($ret,0,-1);
      $afterPoint = $afterPoint-1;
    }
  }
  if(substr($ret,-1)==$decPoint) {$ret = substr($ret,0,-1);}
  return $ret;
}
```

up
down
0

divinity76 at gmail dot com ¶

2 years ago

if you want to benchmark all costs for 5 seconds:

```
<?php
set_time_limit(5);
for($cost=4;;++$cost){
        $start=microtime(true);
        password_hash("test", PASSWORD_BCRYPT, ["cost" => $cost]);
        $end=microtime(true);
        echo "cost {$cost}: ".(int)(($end-$start)*1000)."ms -
".number_format($end-$start,4)."s\n";
}
```

```
?>

on my laptop rolling "Intel Core i7-8565U CPU @ 1.80GHz"  it prints:

$ php foo.php
cost 4: 1ms - 0.0010s
cost 5: 2ms - 0.0022s
cost 6: 3ms - 0.0038s
cost 7: 6ms - 0.0069s
cost 8: 14ms - 0.0147s
cost 9: 25ms - 0.0254s
cost 10: 55ms - 0.0554s
cost 11: 103ms - 0.1040s
cost 12: 184ms - 0.1848s
cost 13: 367ms - 0.3676s
cost 14: 737ms - 0.7379s
cost 15: 1881ms - 1.8810s

(with ms meaning milliseconds and s meaning seconds)
```

up
down
0

mail at igor dot vodka ¶
3 years ago

```
Please be careful with stm555 at hotmail dot com's solution.
If you pass some little negative number (-1 < $number < 0) such as -0.01, the integer part would
be converted to 0, so that the sign is eventually lost.
Here is a fixed version:

<?php
      function number_format_unlimited_precision($number,$decimal = '.')
      {
            $broken_number = explode('.', abs($number));
            $sign = $number < 0 ? '-' : '';
            return $sign.number_format($broken_number[0]).$decimal.$broken_number[1];
      }
?>
```

up
down
1

Svein Tjonndal (sveint at yahoo dot com) ¶
18 years ago

```
If you use space as a separator, it will break on that space in HTML tables...

Furthermore, number_format doesn't like ' ' as a fourth parameter. I wrote the following
function to display the numbers in an HTML table.

  function numberfix($number)
  {
    $number = number_format($number,0,","," ");
    return str_replace(" ", " ", $number);
  }

For use in:
<table><tr><td><?php echo $number; ?></td></tr></table>
```

up
down
0

mobi dot lenoe at gmail dot com ¶
9 years ago

I'd like to comment to the old notes of "stm555" and "woodynadobhar".
They wrote about "number_format_unlimited_precision()".
I guess many of us need that kind of function, which is the almost same function as number_format but don't round a number.

Does Anyone know any new solution in a recent PHP version?

...
If no, how about the following function? (I fixed somethings like bugs of the function in the old comment.)

```
<?php
function number_format_unchanged_precision($number, $dec_point='.', $thousands_sep=','){
    if($dec_point==$thousands_sep){
        trigger_error('2 parameters for ' . __METHOD__ . '() have the same value, that is "' . $dec_point . '" for $dec_point and $thousands_sep', E_USER_WARNING);
        // It corresponds "PHP Warning:  Wrong parameter count for number_format()", which occurs when you use $dec_point without $thousands_sep to number_format().
    }
    if(preg_match('{\.\d+}', $number, $matches)===1){
        $decimals = strlen($matches[0]) - 1;
    }else{
        $decimals = 0;
    }
    return number_format($number, $decimals, $dec_point, $thousands_sep);
}
var_dump(number_format_unchanged_precision(1234.5678, ',', '.'));
var_dump(number_format_unchanged_precision(1234.5678, ','));
var_dump(number_format_unchanged_precision(12345678));
var_dump(number_format_unchanged_precision(-0.5678, ',', '.'));  // It occurred a bug with the function in the old comment.
?>
output is:
    string(10) "1.234,5678"
    PHP Warning:  2 parameters for number_format_unchanged_precision() have the same value, that is "," for $dec_point and $thousands_sep in...
    string(10) "1,234,5678"
    string(10) "12,345,678"
    string(7) "-0,5678"
```

up
down
0

IMSoP ¶
13 years ago

I'm not sure if this is the right place anyway, but "ben at last dot fm"'s ordinal function can be simplified further by removing the redundant "floor" (the result of floor is still a float, it's the "%" that's converting to int) and outer switch.

Note that this version also returns the number with the suffix on the end, not just the suffix.

```
<?php
function ordinal($num)
{
    // Special case "teenth"
    if ( ($num / 10) % 10 != 1 )
    {
        // Handle 1st, 2nd, 3rd
```

```
        switch( $num % 10 )
        {
            case 1: return $num . 'st';
            case 2: return $num . 'nd';
            case 3: return $num . 'rd';
        }
    }
    // Everything else is "nth"
    return $num . 'th';
}
?>
```

up
down
0

Barbara ¶

13 years ago

I was looking for a SIMPLE way to format currency and account for negative values while not losing the calculation properties of my number. Here's my function - it's not rocket science, but maybe can help someone along the way.

```
<?php
function wims_currency($number) {
   if ($number < 0) {
     $print_number = "($ " . str_replace('-', '', number_format ($number, 2, ".", ",")) . ")";
    } else {
     $print_number = "$ " .  number_format ($number, 2, ".", ",") ;
   }
   return $print_number;
}
?>
```

Sample use:

```
<?php
$pur_po_total =  ($pur_po_total + $pur_item_total);
$print_pur_po_total = wims_currency($pur_po_total);
?>
```

Returns (for example) $ 44,561.00 or, if a negative ($ 407,250.00)

This way, I use my 1st variable for calculations and my 2nd variable for output. I'm sure there are better ways to do it, but this got me back on track.

up
down
0

gabrielu at gmail dot com ¶

14 years ago

Using the number_format I'm having some unexpected results. 30% of 14.95 (14.95 * .3) = 4.485. Now 4.485 rounded to two decimal places should give me 4.49.

Example:

```
<?php
echo number_format(14.95 * .3, 2, '.', '') . "\n";
echo number_format(4.485, 2, '.', '') . "\n";
?>
```

Unexpected Results:

```
4.48
4.49
```

up
down
0
uliciadrian01 at yahoo dot com ¶
15 years ago
A simple funtion to format american dollars.

```
<?php
function formatMoney($money) {
    if($money<1) {
        $money='¢'.$money*100;
    }
    else {
        $dollars=intval($money);
        $cents=$money-$dollars;
        $cents=$cents*100;
        $money='$'.$dollars.' and ¢'.$cents;
    }
    return $money;
}
echo formatmoney('52.342');
?>
This will output: "   $52 and ¢34.2  ".
```

up
down
0
webmaster at WWW.ELLESSEWEB.NET ¶
15 years ago
This is a simple and useful function to convert a byte number in a KB or MB:

```
<?
function filesize_format ($bytes) {
  $bytes=(float)$bytes;
  if ($bytes<1024){
  $numero=number_format($bytes, 0, ',', '.')." Byte";
  return $numero;
  }
  if ($bytes<1048576){
      $numero=number_format($bytes/1024, 2, ',', '.')." KByte";
  return $numero;
  }
  if ($bytes>=1048576){
      $numero=number_format($bytes/1048576, 2, ',', '.')." MByte";
  return $numero;
  }
}
?>
```

up
down
0
Jeroen de Bruijn [NL] ¶
17 years ago
If you want to display a number ending with ,- (like 200,-) when there are no decimal characters and display the decimals when there are decimal characters i use:

```
function DisplayDouble($value)
  {
  list($whole, $decimals) = split ('[.,]', $value, 2);
```

```
  if (intval($decimals) > 0)
    return number_format($value,2,".","");
  else
    return number_format($value,0,".","") .",-";
  }
```

up
down
0

marc dot vanwoerkom at fernuni-hagen dot de ¶
17 years ago

See also the documentation for localeconv, which will provide values for decimal point and thousands separator from the C standard library.

Of course localeconv features many more locale information, like indicating to put the negative sign behind the value for some locale settings which can't be used to customize present number_format.

up
down
-1

Ali Okan YKSEL ¶
12 years ago

function formats numbers of datetime type,

```
<?php
$_GET["zaman"]="1983-8-28 5:5:5";

function _parseDatetimeToList($datetimeStr) { //datetime format: Y-m-d H-i-s
    $datetimeArray=explode(" ", $datetimeStr);
    $dateArray=explode("-",$datetimeArray[0]);
    $year=str_pad($dateArray[0], 2, "0", STR_PAD_LEFT);
    $month=str_pad($dateArray[1], 2, "0", STR_PAD_LEFT);
    $day=str_pad($dateArray[2], 2, "0", STR_PAD_LEFT);
    $timeArray=explode(":",$datetimeArray[1]);
    $hour=str_pad($timeArray[0], 2, "0", STR_PAD_LEFT);
    $minute=str_pad($timeArray[1], 2, "0", STR_PAD_LEFT);
    $second=str_pad($timeArray[2], 2, "0", STR_PAD_LEFT);
    return array($year, $month, $day, $hour, $minute, $second);
}

list($year, $month, $day, $hour, $minute, $second) = _parseDatetimeToList($_GET["zaman"]);  //
1983-1-28 5:5:5
?>
```

up
down
-1

samuelpeixoto at gmail dot com ¶
13 years ago

Exemplo: Example:

```
<?php
$number = 1234567.896;
echo '1: '.number_format($number, 2, ',', '').'<br>';
echo '2: '.number_format($number, 2, '.', '').'<br>';
echo '3: '.number_format($number, 2, ',', '.').'<br>';
echo '4: '.number_format($number, 2, '.', ',').'<br>';
echo '5: '.number_format($number, 2, ',', ' ').'<br>';
echo '6: '.number_format($number, 2, ',', "'").'<br>';
echo '7: '.number_format($number, 2, '', '').'<br>';
```

```
?>

Resultado: Result:

1: 1234567,90   -> Decimal separado por ,
2: 1234567.90   -> Decimal separado por .
3: 1.234.567,90 -> Moeda Brasil, Alemanha
4: 1,234,567.90 -> Inglês, USA
5: 1 234 567,90 -> França
6: 1'234'567,90 -> Suíça
7: 123456790    -> Sem decimal
```

up
down
-1
woodynadobhar at hotmail dot com ¶
17 years ago

What do you do if some of your numbers have decimal places, and some don't? You can switch between functions, but if you're building it in a loop, that's not a good solution. Instead, we have the same as below, with a slight change:

```
function number_format_unlimited_precision($number,$decimal = '.'){
    $broken_number = explode($decimal,$number);
    if($broken_number[1]==0){
        return number_format($broken_number[0]);
    }else{
        return number_format($broken_number[0]).$decimal.$broken_number[1];
    };
};
```

up
down
-2
dipu dot ashok dot 17 at gmail dot com ¶
12 years ago

function to convert numbers to words
indian: thousand,lakh,crore
Note: function can only convert nos upto 99 crores

```
<?php
$words = array('0'=> '' ,'1'=> 'one' ,'2'=> 'two' ,'3' => 'three','4' => 'four','5' => 'five','6'
=> 'six','7' => 'seven','8' => 'eight','9' => 'nine','10' => 'ten','11' => 'eleven','12' =>
'twelve','13' => 'thirteen','14' => 'fouteen','15' => 'fifteen','16' => 'sixteen','17' =>
'seventeen','18' => 'eighteen','19' => 'nineteen','20' => 'twenty','30' => 'thirty','40' =>
'fourty','50' => 'fifty','60' => 'sixty','70' => 'seventy','80' => 'eighty','90' => 'ninty','100'
=> 'hundred &','1000' => 'thousand','100000' => 'lakh','10000000' => 'crore');
function no_to_words($no)
{    global $words;
    if($no == 0)
        return ' ';
    else {          $novalue='';$highno=$no;$remainno=0;$value=100;$value1=1000;
            while($no>=100)    {
                if(($value <= $no) &&($no  < $value1))    {
                $novalue=$words["$value"];
                $highno = (int)($no/$value);
                $remainno = $no % $value;
                break;
                }
                $value= $value1;
                $value1 = $value * 100;
```

```
            }
          if(array_key_exists("$highno",$words))
              return $words["$highno"]." ".$novalue." ".no_to_words($remainno);
          else {
             $unit=$highno%10;
             $ten =(int)($highno/10)*10;
             return $words["$ten"]." ".$words["$unit"]." ".$novalue." ".no_to_words($remainno);
           }
    }
}
echo no_to_words(999978987);

?>
```

+ add a note

- Funciones de strings
  - addcslashes
  - addslashes
  - bin2hex
  - chop
  - chr
  - chunk_split
  - convert_uudecode
  - convert_uuencode
  - count_chars
  - crc32
  - crypt
  - echo
  - explode
  - fprintf
  - get_html_translation_table
  - hebrev
  - hex2bin
  - html_entity_decode
  - htmlentities
  - htmlspecialchars_decode
  - htmlspecialchars
  - implode
  - join
  - lcfirst
  - levenshtein
  - localeconv
  - ltrim
  - md5_file
  - md5
  - metaphone
  - money_format
  - nl_langinfo
  - nl2br
  - number_format
  - ord
  - parse_str
  - print
  - printf
  - quoted_printable_decode
  - quoted_printable_encode
  - quotemeta
  - rtrim

- setlocale
- sha1_file
- sha1
- similar_text
- soundex
- sprintf
- sscanf
- str_contains
- str_ends_with
- str_getcsv
- str_ireplace
- str_pad
- str_repeat
- str_replace
- str_rot13
- str_shuffle
- str_split
- str_starts_with
- str_word_count
- strcasecmp
- strchr
- strcmp
- strcoll
- strcspn
- strip_tags
- stripcslashes
- stripos
- stripslashes
- stristr
- strlen
- strnatcasecmp
- strnatcmp
- strncasecmp
- strncmp
- strpbrk
- strpos
- strrchr
- strrev
- strripos
- strrpos
- strspn
- strstr
- strtok
- strtolower
- strtoupper
- strtr
- substr_compare
- substr_count
- substr_replace
- substr
- trim
- ucfirst
- ucwords
- utf8_decode
- utf8_encode
- vfprintf
- vprintf
- vsprintf

- wordwrap
- Deprecated
  - convert_cyr_string
  - hebrevc