parse_str »
« number_format

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

Change language: Spanish

Submit a Pull Request Report a Bug

ord

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

ord — Convierte el primer byte de un string a un valor entre 0 y 255

Descripción ¶

ord(string `$string`): int

Interpreta el valor binario del primer byte de `string` como un entero sin signo entre 0 y 255.

Si el string está en codificación de byte único, como ASCII, ISO-8859 o Windows 1252, esto es equivalente a devolver la posición del carácter de la tabla de correspondencia del conjunto de caracteres. Sin embargo, observe que esta función no conoce la codificación del string, por lo que nunca identificará un punto de código Unicode en una codificación multibyte tal como UTF-8 o UTF-16.

Esta función complementa a chr().

Parámetros ¶

`string`

Un carácter.

Valores devueltos ¶

Un entero entre 0 y 255.

Ejemplos ¶

Ejemplo #1 Ejemplo de ord()

```
<?php
$str = "\n";
if (ord($str) == 10) {
    echo "El primer caracter de \$str es un salto de linea.\n";
}
?>
```

Ejemplo #2 Examinar los bytes individuales de un string UTF-8

```
<?php
declare(encoding='UTF-8');
```

```
$str = "🐘";
for ( $pos=0; $pos < strlen($str); $pos ++ ) {
 $byte = substr($str, $pos);
 echo 'El byte ' . $pos . ' de $str tiene el valor ' . ord($byte) . PHP_EOL;
}
?>
```

El resultado del ejemplo sería:

El byte 0 de $str tiene el valor 240
El byte 1 de $str tiene el valor 159
El byte 2 de $str tiene el valor 144
El byte 3 de $str tiene el valor 152

Ver también ¶

- chr() - Devuelve un caracter específico
- An » ASCII-table
- mb_ord() - Get Unicode code point of character

+ add a note

User Contributed Notes 6 notes

up
down
45
arglanir+phpnet at gmail dot com ¶
10 years ago

As ord() doesn't work with utf-8, and if you do not have access to mb_* functions, the following function will work well:

```
<?php
function ordutf8($string, &$offset) {
    $code = ord(substr($string, $offset,1));
    if ($code >= 128) {        //otherwise 0xxxxxxx
        if ($code < 224) $bytesnumber = 2;                //110xxxxx
        else if ($code < 240) $bytesnumber = 3;        //1110xxxx
        else if ($code < 248) $bytesnumber = 4;    //11110xxx
        $codetemp = $code - 192 - ($bytesnumber > 2 ? 32 : 0) - ($bytesnumber > 3 ? 16 : 0);
        for ($i = 2; $i <= $bytesnumber; $i++) {
            $offset ++;
            $code2 = ord(substr($string, $offset, 1)) - 128;        //10xxxxxx
            $codetemp = $codetemp*64 + $code2;
        }
        $code = $codetemp;
    }
    $offset += 1;
    if ($offset >= strlen($string)) $offset = -1;
    return $code;
}
?>
```

$offset is a reference, as it is not easy to split a utf-8 char-by-char. Useful to iterate on a string:

```
<?php
$text = "abcàêß€abc";
$offset = 0;
```

```
while ($offset >= 0) {
    echo $offset.": ".ordutf8($text, $offset)."\n";
}
/* returns:
0: 97
1: 98
2: 99
3: 224
5: 234
7: 223
9: 8364
12: 97
13: 98
14: 99
*/
?>
```

Feel free to adapt my code to fit your needs.

up
down
17

rowan dot collins at cwtdigital dot com ¶
9 years ago

Regarding character sets, and whether or not this is "ASCII". Firstly, there is no such thing as "8-bit ASCII", so if it were ASCII it would only ever return integers up to 127. 8-bit ASCII-compatible encodings include the ISO 8859 family of encodings, which map various common characters to the values from 128 to 255. UTF-8 is also designed so that characters representable in 7-bit ASCII are coded the same; byte values higher than 127 in a UTF-8 string represent the beginning of a multi-byte character.

In fact, like most of PHP's string functions, this function isn't doing anything to do with character encoding at all - it is just interpreting a binary byte from a string as an unsigned integer. That is, ord(chr(200)) will always return 200, but what character chr(200) *means* will vary depending on what character encoding it is *interpreted* as part of (e.g. during display).

A technically correct description would be "Returns an integer representation of the first byte of a string, from 0 to 255. For single-byte encodings such as (7-bit) ASCII and the ISO 8859 family, this will correspond to the first character, and will be the position of that character in the encoding's mapping table. For multi-byte encodings, such as UTF-8 or UTF-16, the byte may not represent a complete character."

The link to asciitable.com should also be replaced by one which explains what character encoding it is displaying, as "Extended ASCII" is an ambiguous and misleading name.

up
down
4

paco at olecode dot com ¶
2 years ago

this function convert UTF-8 string to RTF code string. I am using code of v0rbiz at yahoo dot com, thanks!!!

```
function cadena_rtf($txt)
   {
      $result = null;

      for ($pos = 0; $pos < mb_strlen($txt); $pos++) {

         $char = mb_substr($txt, $pos, 1);
```

```
        if (!preg_match("/[A-Za-z1-9,.]/", $char)) {
           //unicode ord real!!!
           $k   = mb_convert_encoding($char, 'UCS-2LE', 'UTF-8');
           $k1  = ord(substr($k, 0, 1));
           $k2  = ord(substr($k, 1, 1));
           $ord = $k2 * 256 + $k1;

           if ($ord > 255) {
              $result .= '\uc1\u' . $ord . '*';
           } elseif ($ord > 32768) {
              $result .= '\uc1\u' . ($ord - 65535) . '*';
           } else {
              $result .= "\\'" . dechex($ord);
           }
        } else {
           $result .= $char;
        }
     }
     return $result;
  }
```

up
down
9

v0rbiz at yahoo dot com ¶

18 years ago

```
I did not found a unicode/multibyte capable 'ord' function, so...

<?php
function uniord($u) {
    $k = mb_convert_encoding($u, 'UCS-2LE', 'UTF-8');
    $k1 = ord(substr($k, 0, 1));
    $k2 = ord(substr($k, 1, 1));
    return $k2 * 256 + $k1;
}
?>
```

up
down
2

Noname ¶

9 months ago

```
<?php

declare (encoding='UTF-8');

$animalsstr = '🐀🐁🐂🐃🐄🐅🐆🐇🐈🐉🐊🐋🐌🐍🐎🐏🐐🐑🐒🐓🐔🐕🐖🐗🐘🐙'
        . '🐚🐛🐜🐝🐞🐟🐠🐡🐢🐣🐤🐥🐦🐧🐨🐩🐪🐫🐬🐭🐮🐯🐰🐱🐲🐳🐴🐵'
        . '🐶🐷🐸🐹🐺🐻🐼🐽🐾🐿';

$animals = mb_str_split($animalsstr);
foreach ($animals as $animal) {
    for ($pos = 0; $pos < strlen($animal); $pos++) {
        $byte = substr($animal, $pos);
        echo "Byte $pos of $animal has value " . ord($byte) . PHP_EOL;
    }
}

?>
```

up

down
-3
Anonymous ¶
1 year ago

For anyone who's looking to convert full strings to map and back it's pretty simple but takes some getting used to...the code below saves an hour of scrounging codes for beginners like myself.

```
function var2map($a) {
    $b='';
    $c=strlen($a);
    for($i=0; $i<$c; ++$i) {
        $d=ord(substr($a,$i,1));
        if($d<10) {
            $e='00'.$d;
        } else {
            if($d<100) {
                $e='0'.$d;
            } else {
                $e=$d;
            }
        }
        if($b=='') {
            $b=$e;
        } else {
            $b=$b.$e;
        }
    }
    return $b;
}

function map2var($a) {
    $b='';
    $c=strlen($a) / 3;
    for($i=0; $i<$c; ++$i) {
        $d=chr(substr($a,$i*3,3));
        if($b=='') {
            $b=$d;
        } else {
            $b=$b.$d;
        }
    }
    return $b;
}
```

+ add a note

- 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
- strpos
- stripslashes