Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

g h

Goto homepage
g s
Goto search
(current page)
/
Focus search box

getdate »
« date_timezone_set

- PHP Manual
- Function Reference
- Date and Time Related Extensions
- Date/Time
- Date/Time Functions

Change language: English

Edit Report a Bug

date

(PHP 4, PHP 5, PHP 7)

date — Format a local time/date

Description ¶

date (string `$format` [, int `$timestamp` = time()]) : string

Returns a string formatted according to the given format string using the given integer `timestamp` or the current time if no timestamp is given. In other words, `timestamp` is optional and defaults to the value of time().

Parameters ¶

`format`

The format of the outputted date string. See the formatting options below. There are also several predefined date constants that may be used instead, so for example **`DATE_RSS`** contains the format string *'D, d M Y H:i:s'*.

The following characters are recognized in the `format` parameter string

`format` character	Description	Example returned values
Day	---	---
d	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through *Sun*
j	Day of the month without leading zeros	*1* to *31*
l (lowercase 'L')	A full textual representation of the day of the week	*Sunday* through *Saturday*

`format` character	Description	Example returned values
N	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	*1* (for Monday) through *7* (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
w	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
z	The day of the year (starting from 0)	*0* through *365*
Week	---	---
W	ISO-8601 week number of year, weeks starting on Monday	Example: *42* (the 42nd week in the year)
Month	---	---
F	A full textual representation of a month, such as January or March	*January* through *December*
m	Numeric representation of a month, with leading zeros	*01* through *12*
M	A short textual representation of a month, three letters	*Jan* through *Dec*
n	Numeric representation of a month, without leading zeros	*1* through *12*
t	Number of days in the given month	*28* through *31*
Year	---	---
L	Whether it's a leap year	*1* if it is a leap year, *0* otherwise.
o	ISO-8601 week-numbering year. This has the same value as *Y*, except that if the ISO week number (*W*) belongs to the previous or next year, that year is used instead. (added in PHP 5.1.0)	Examples: *1999* or *2003*
Y	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
y	A two digit representation of a year	Examples: *99* or *03*
Time	---	---
a	Lowercase Ante meridiem and Post meridiem	*am* or *pm*
A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*
B	Swatch Internet time	*000* through *999*
g	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
h	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
i	Minutes with leading zeros	*00* to *59*
s	Seconds with leading zeros	*00* through *59*
u	Microseconds (added in PHP 5.2.2). Note that **date()** will always generate *000000* since it takes an integer parameter, whereas DateTime::format() does support microseconds if DateTime was created with microseconds.	Example: *654321*
v	Milliseconds (added in PHP 7.0.0). Same note applies as for *u*.	Example: *654*
Timezone	---	---

format character	Description	Example returned values
e	Timezone identifier (added in PHP 5.1.0)	Examples: *UTC*, *GMT*, *Atlantic/Azores*
I (capital i)	Whether or not the date is in daylight saving time	*1* if Daylight Saving Time, *0* otherwise.
O	Difference to Greenwich time (GMT) without colon between hours and minutes	Example: *+0200*
P	Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3)	Example: *+02:00*
T	Timezone abbreviation	Examples: *EST*, *MDT* ...
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	*-43200* through *50400*
Full Date/Time	---	---
c	ISO 8601 date (added in PHP 5)	2004-02-12T15:19:21+00:00
r	» RFC 2822 formatted date	Example: *Thu, 21 Dec 2000 16:01:07 +0200*
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	See also time()

Unrecognized characters in the format string will be printed as-is. The *Z* format will always return *0* when using gmdate().

> **Note**:
>
> Since this function only accepts integer timestamps the *u* format character is only useful when using the date_format() function with user based timestamps created with date_create().

`timestamp`

The optional `timestamp` parameter is an integer Unix timestamp that defaults to the current local time if a `timestamp` is not given. In other words, it defaults to the value of time().

Return Values ¶

Returns a formatted date string. If a non-numeric value is used for `timestamp`, **`FALSE`** is returned and an **`E_WARNING`** level error is emitted.

Errors/Exceptions ¶

Every call to a date/time function will generate a **`E_NOTICE`** if the time zone is not valid, and/or a **`E_STRICT`** or **`E_WARNING`** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

Changelog ¶

Version	Description
5.1.1	There are useful constants of standard date/time formats that can be used to specify the `format` parameter.
5.1.0	The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer). However, before PHP 5.1.0 this range was limited from 01-01-1970 to 19-01-2038 on some systems (e.g. Windows).
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples ¶

Example #1 date() examples

```
<?php
// set the default timezone to use. Available since PHP 5.1
date_default_timezone_set('UTC');


// Prints something like: Monday
echo date("l");

// Prints something like: Monday 8th of August 2005 03:12:46 PM
echo date('l jS \of F Y h:i:s A');

// Prints: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

/* use the constants in the format parameter */
// prints something like: Wed, 25 Sep 2013 15:28:57 -0700
echo date(DATE_RFC2822);

// prints something like: 2000-07-01T00:00:00+00:00
echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));
?>
```

You can prevent a recognized character in the format string from being expanded by escaping it with a preceding backslash. If the character with a backslash is already a special sequence, you may need to also escape the backslash.

Example #2 Escaping characters in date()

```
<?php
// prints something like: Wednesday the 15th
echo date('l \t\h\e jS');
?>
```

It is possible to use **date()** and mktime() together to find dates in the future or the past.

Example #3 date() and mktime() example

```
<?php
$tomorrow  = mktime(0, 0, 0, date("m")  , date("d")+1, date("Y"));
```

```
$lastmonth = mktime(0, 0, 0, date("m")-1, date("d"),   date("Y"));
$nextyear  = mktime(0, 0, 0, date("m"),   date("d"),   date("Y")+1);
?>
```

> **Note**:
>
> This can be more reliable than simply adding or subtracting the number of seconds in a day or month to a timestamp because of daylight saving time.

Some examples of **date()** formatting. Note that you should escape any other characters, as any which currently have a special meaning will produce undesirable results, and other characters may be assigned meaning in future PHP versions. When escaping, be sure to use single quotes to prevent characters like \n from becoming newlines.

Example #4 date() Formatting

```
<?php
// Assuming today is March 10th, 2001, 5:16:18 pm, and that we are in the
// Mountain Standard Time (MST) Time Zone

$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day');     // 05-16-18, 10-03-01, 1631 1618 6 Satpm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // it is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 17:16:18 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:18 m is month
$today = date("H:i:s");                         // 17:16:18
$today = date("Y-m-d H:i:s");                   // 2001-03-10 17:16:18 (the MySQL DATETIME format)
?>
```

To format dates in other languages, you should use the setlocale() and strftime() functions instead of **date()**.

Notes ¶

> **Note**:
>
> To generate a timestamp from a string representation of the date, you may be able to use strtotime(). Additionally, some databases have functions to convert their date formats into timestamps (such as MySQL's » UNIX_TIMESTAMP function).

Tip

Timestamp of the start of the request is available in *$_SERVER['REQUEST_TIME']* since PHP 5.1.

See Also ¶

- gmdate() - Format a GMT/UTC date/time
- idate() - Format a local time/date as integer
- getdate() - Get date/time information
- getlastmod() - Gets time of last page modification
- mktime() - Get Unix timestamp for a date
- strftime() - Format a local time/date according to locale settings

- time() - Return current Unix timestamp
- strtotime() - Parse about any English textual datetime description into a Unix timestamp
- Predefined DateTime Constants

add a note

User Contributed Notes 18 notes

up
down
106
Jimmy ¶
8 years ago

```
Things to be aware of when using week numbers with years.

<?php
echo date("YW", strtotime("2011-01-07")); // gives 201101
echo date("YW", strtotime("2011-12-31")); // gives 201152
echo date("YW", strtotime("2011-01-01")); // gives 201152 too
?>

BUT

<?php
echo date("oW", strtotime("2011-01-07")); // gives 201101
echo date("oW", strtotime("2011-12-31")); // gives 201152
echo date("oW", strtotime("2011-01-01")); // gives 201052 (Year is different than previous example)
?>

Reason:
Y is year from the date
o is ISO-8601 year number
W is ISO-8601 week number of year

Conclusion:
if using 'W' for the week number use 'o' for the year.
```

up
down
17
adityabhai at gmail dot com ¶
6 years ago

```
For Microseconds, we can get by following:

echo date('Ymd His'.substr((string)microtime(), 1, 8).' e');

Thought, it might be useful to someone !
```

up
down
9
Charlie ¶
4 years ago

```
For HTML5 datetime-local HTML input controls (http://www.w3.org/TR/html-markup/input.datetime-local.html) use format example: 1996-12-19T16:39:57
```

To generate this, escape the 'T', as shown below:

```
<?php
date('Y-m-d\TH:i:s');
?>
```

up
down
11
Anonymous ¶
5 years ago

It's common for us to overthink the complexity of date/time calculations and underthink the power and flexibility of PHP's built-in functions. Consider http://php.net/manual/en/function.date.php#108613

```
<?php
function get_time_string($seconds)
{
    return date('H:i:s', strtotime("2000-01-01 + $seconds SECONDS"));
}
```

up
down
15
FiraSEO ¶
6 years ago

this how you make an HTML5 <time> tag correctly

```
<?php

echo '<time datetime="'.date('c').'">'.date('Y - m - d').'</time>';

?>
```

in the "datetime" attribute you should put a machine-readable value which represent time , the best value is a full time/date with ISO 8601 (date('c')) ,,, the attr will be hidden from users

and it doesn't really matter what you put as a shown value to the user,, any date/time format is okay !

This is very good for SEO especially search engines like Google .

up
down
7
Anonymous ¶
3 years ago

If timestamp is a string, date converts it to an integer in a possibly unexpected way:

```
<?php
echo (int)'0x10'; //0
echo intval('0x10'); //0
echo date('s', '0x10'); //gives 16
//however, no octal conversion:
echo date('s', '010'); //gives 10
?>
```

(PHP 5.6.16)

up
down
6
david dot thomas at elliott-thomas dot com dot au ¶
3 years ago
Prior to PHP 5.6.23, Relative Formats for the start of the week aligned with PHP's (0=Sunday,6=Saturday). Since 5.6.23, Relative Formats for the start of the week align with ISO-8601 (1=Monday,7=Sunday). (http://php.net/manual/en/datetime.formats.relative.php)

This can produce different, and seemingly incorrect, results depending on your PHP version and your choice of 'w' or 'N' for the Numeric representation of the day of the week:

```
<?php
echo "Today is Sun 2 Oct 2016, day ",date('w',strtotime('2016-10-02'))," of this week. ";
echo "Day ",date('w',strtotime('2016-10-02 Monday next week'))," of next week is ",date('d M
Y',strtotime('2016-10-02 Monday next week')),"<br />";

echo "Today is Sun 2 Oct 2016, day ",date('N',strtotime('2016-10-02'))," of this week. ";
echo "Day ",date('w',strtotime('2016-10-02 Monday next week'))," of next week is ",date('d M
Y',strtotime('2016-10-02 Monday next week'));
?>
```

Prior to PHP 5.6.23, this results in:

```
Today is Sun 2 Oct 2016, day 0 of this week. Day 1 of next week is 10 Oct 2016
Today is Sun 2 Oct 2016, day 7 of this week. Day 1 of next week is 10 Oct 2016
```

Since PHP 5.6.23, this results in:

```
Today is Sun 2 Oct 2016, day 0 of this week. Day 1 of next week is 03 Oct 2016
Today is Sun 2 Oct 2016, day 7 of this week. Day 1 of next week is 03 Oct 2016
```

up
down
5
rc at macshot dot de ¶
2 years ago
At least in PHP 5.5.38 date('j.n.Y', 2222222222) gives a result of 2.6.2040.

So date is not longer limited to the minimum and maximum values for a 32-bit signed integer as timestamp.

up
down
10
matthew dot hotchen at worldfirst dot com ¶
5 years ago
FYI: there's a list of constants with predefined formats on the DateTime object, for example instead of outputting ISO 8601 dates with:

```
<?php
echo date('c');
?>
```

or

```
<?php
echo date('Y-m-d\TH:i:sO');
?>
```

You can use

```
<?php
echo date(DateTime::ISO8601);
?>
```

instead, which is much easier to read.

up
down
11

ivijan dot stefan at gmail dot com ¶
4 years ago

If you have a problem with the different time zone, this is the solution for that.

```
<?php
// first line of PHP
$defaultTimeZone='UTC';
if(date_default_timezone_get()!=$defaultTimeZone)) date_default_timezone_set($defaultTimeZone);

// somewhere in the code
function _date($format="r", $timestamp=false, $timezone=false)
{
    $userTimezone = new DateTimeZone(!empty($timezone) ? $timezone : 'GMT');
    $gmtTimezone = new DateTimeZone('GMT');
    $myDateTime = new DateTime(($timestamp!=false?date("r",(int)$timestamp):date("r")),
$gmtTimezone);
    $offset = $userTimezone->getOffset($myDateTime);
    return date($format, ($timestamp!=false?(int)$timestamp:$myDateTime->format('U')) + $offset);
}

/* Example */
echo 'System Date/Time: '.date("Y-m-d | h:i:sa").'<br>';
echo 'New York Date/Time: '._date("Y-m-d | h:i:sa", false, 'America/New_York').'<br>';
echo 'Belgrade Date/Time: '._date("Y-m-d | h:i:sa", false, 'Europe/Belgrade').'<br>';
echo 'Belgrade Date/Time: '._date("Y-m-d | h:i:sa", 514640700, 'Europe/Belgrade').'<br>';
?>
```

This is the best and fastest solution for this problem. Working almost identical to date() function only as a supplement has the time zone option.

up
down
1

mirco dot babin at gmail dot com ¶
1 year ago

One important thing you should remember is that the timestamp value returned by time() is time-zone agnostic and gets the number of seconds since 1 January 1970 at 00:00:00 UTC. This means that at a particular point in time, this function will return the same value in the US, Europe, India, Japan, ...

date() will format a time-zone agnostic timestamp according to the default timezone set with date_default_timezone_set(...). Local time. If you want to output as UTC time use:

```
<?php
function dateUTC($format, $timestamp = null)
{
    if ($timestamp === null) $timestamp = time();

    $tz = date_default_timezone_get();
    date_default_timezone_set('UTC');

    $result = date($format, $timestamp);

    date_default_timezone_set($tz);
    return $result;
}
/>
```

up
down
5

SpikeDaCruz ¶

13 years ago

The following function will return the date (on the Gregorian calendar) for Orthodox Easter (Pascha). Note that incorrect results will be returned for years less than 1601 or greater than 2399. This is because the Julian calendar (from which the Easter date is calculated) deviates from the Gregorian by one day for each century-year that is NOT a leap-year, i.e. the century is divisible by 4 but not by 10. (In the old Julian reckoning, EVERY 4th year was a leap-year.)

This algorithm was first proposed by the mathematician/physicist Gauss. Its complexity derives from the fact that the calculation is based on a combination of solar and lunar calendars.

```
<?php
function getOrthodoxEaster($date){
  /*
   Takes any Gregorian date and returns the Gregorian
   date of Orthodox Easter for that year.
  */
  $year = date("Y", $date);
  $r1 = $year % 19;
  $r2 = $year % 4;
  $r3 = $year % 7;
  $ra = 19 * $r1 + 16;
  $r4 = $ra % 30;
  $rb = 2 * $r2 + 4 * $r3 + 6 * $r4;
  $r5 = $rb % 7;
  $rc = $r4 + $r5;
  //Orthodox Easter for this year will fall $rc days after April 3
  return strtotime("3 April $year + $rc days");
}
?>
```

up
down
0

bruslbn at gmail dot com ¶

1 year ago

In order to define leap year you must considre not only that year can be divide by 4!

The correct alghoritm is:

if (year is not divisible by 4) then (it is a common year)
else if (year is not divisible by 100) then (it is a leap year)
else if (year is not divisible by 400) then (it is a common year)
else (it is a leap year)

So the code should look like this:

```
if($year%4 == 0 && $year%100 != 0) {
    $leapYear = 1;
} elseif($year%400 == 0) {
    $leapYear = 1;
} else {
    $leapYear = 0;
}
```

up
down
1

Bas Vijfwinkel ¶

7 years ago

Note that some formatting options are different from MySQL.
For example using a 24 hour notation without leading zeros is the option '%G' in PHP but '%k' in MySQL.
When using dynamically generated date formatting string, be careful to generate the correct options for either PHP or MySQL.

up
down
1

ghotinet ¶

9 years ago

Most spreadsheet programs have a rather nice little built-in function called NETWORKDAYS to calculate the number of business days (i.e. Monday-Friday, excluding holidays) between any two given dates. I couldn't find a simple way to do that in PHP, so I threw this together. It replicates the functionality of OpenOffice's NETWORKDAYS function - you give it a start date, an end date, and an array of any holidays you want skipped, and it'll tell you the number of business days (inclusive of the start and end days!) between them.

I've tested it pretty strenuously but date arithmetic is complicated and there's always the possibility I missed something, so please feel free to check my math.

The function could certainly be made much more powerful, to allow you to set different days to be ignored (e.g. "skip all Fridays and Saturdays but include Sundays") or to set up dates that should always be skipped (e.g. "skip July 4th in any year, skip the first Monday in September in any year"). But that's a project for another time.

```
<?php

function networkdays($s, $e, $holidays = array()) {
    // If the start and end dates are given in the wrong order, flip them.
    if ($s > $e)
        return networkdays($e, $s, $holidays);

    // Find the ISO-8601 day of the week for the two dates.
```

```
    $sd = date("N", $s);
    $ed = date("N", $e);

    // Find the number of weeks between the dates.
    $w = floor(($e - $s)/(86400*7));    # Divide the difference in the two times by seven days to get
the number of weeks.
    if ($ed >= $sd) { $w--; }        # If the end date falls on the same day of the week or a later
day of the week than the start date, subtract a week.

    // Calculate net working days.
    $nwd = max(6 - $sd, 0);    # If the start day is Saturday or Sunday, add zero, otherwise add six
minus the weekday number.
    $nwd += min($ed, 5);    # If the end day is Saturday or Sunday, add five, otherwise add the
weekday number.
    $nwd += $w * 5;        # Add five days for each week in between.

    // Iterate through the array of holidays. For each holiday between the start and end dates that
isn't a Saturday or a Sunday, remove one day.
    foreach ($holidays as $h) {
        $h = strtotime($h);
        if ($h > $s && $h < $e && date("N", $h) < 6)
            $nwd--;
    }

    return $nwd;
}

$start = strtotime("1 January 2010");
$end = strtotime("13 December 2010");

// Add as many holidays as desired.
$holidays = array();
$holidays[] = "4 July 2010";            // Falls on a Sunday; doesn't affect count
$holidays[] = "6 September 2010";        // Falls on a Monday; reduces count by one

echo networkdays($start, $end, $holidays);    // Returns 246

?>
```

Or, if you just want to know how many work days there are in any given year, here's a quick function for that one:

```
<?php

function workdaysinyear($y) {
    $j1 = mktime(0,0,0,1,1,$y);
    if (date("L", $j1)) {
        if (date("N", $j1) == 6)
            return 260;
        elseif (date("N", $j1) == 5 or date("N", $j1) == 7)
            return 261;
        else
            return 262;
    }
```

```
    else {
        if (date("N", $j1) == 6 or date("N", $j1) == 7)
            return 260;
        else
            return 261;
    }
}

?>
```

up
down
-5

arth dot inbox at gmail dot com ¶
1 year ago

Looks like date('u') is not microseconds, but is positive difference from rest part.

```
php > echo (DateTime::createFromFormat('U.u', '-128649659.999998'))->format('Y-m-d H:i:s.u U.u');
1965-12-03 23:59:01.999998 -128649659.999998
```

`U.u` parsed and formatted same, but means not 1965-12-03 23:59:00.000002.
Other words correct timestamp for example above is (-128649659 + 0.999998).

Less confusing format for it is:

```
php > echo DateTime::createFromFormat('U\+0.u', '-128649660+0.000002')->format('Y-m-d H:i:s.u');
1965-12-03 23:59:00.000002
```

Is that bug or feature?

up
down
-9

bruslbn at gmail dot com ¶
1 year ago

In order to define leap year you must considre not only that year can be divide by 4!

The correct alghoritm is:

```
if (year is not divisible by 4) then (it is a common year)
else if (year is not divisible by 100) then (it is a leap year)
else if (year is not divisible by 400) then (it is a common year)
else (it is a leap year)
```

So the code should look like this:

```
if($year%4 == 0 && $year%100 != 0) {
    $leapYear = 1;
} elseif($year%400 == 0) {
    $leapYear = 1;
} else {
    $leapYear = 0;
}
```

up
down
-11

bruslbn at gmail dot com ¶

1 year ago

```
In order to define leap year you must considre not only that year can be divide by 4!

The correct alghoritm is:

if (year is not divisible by 4) then (it is a common year)
else if (year is not divisible by 100) then (it is a leap year)
else if (year is not divisible by 400) then (it is a common year)
else (it is a leap year)

So the code should look like this:

if($year%4 == 0 && $year%100 != 0) {
    $leapYear = 1;
} elseif($year%400 == 0) {
    $leapYear = 1;
} else {
    $leapYear = 0;
}
```

add a note

- Date/Time Functions
  - checkdate
  - date_add
  - date_create_from_format
  - date_create_immutable_from_format
  - date_create_immutable
  - date_create
  - date_date_set
  - date_default_timezone_get
  - date_default_timezone_set
  - date_diff
  - date_format
  - date_get_last_errors
  - date_interval_create_from_date_string
  - date_interval_format
  - date_isodate_set
  - date_modify
  - date_offset_get
  - date_parse_from_format
  - date_parse
  - date_sub
  - date_sun_info
  - date_sunrise
  - date_sunset
  - date_time_set
  - date_timestamp_get
  - date_timestamp_set
  - date_timezone_get
  - date_timezone_set
  - date
  - getdate
  - gettimeofday