

php



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Errors](#)

[Exceptions](#)

[Generators](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[Using Register Globals](#)

[User Submitted Data](#)

[Magic Quotes](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)
[Using remote files](#)
[Connection handling](#)
[Persistent Database Connections](#)
[Safe Mode](#)
[Command line usage](#)
[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Credit Card Processing](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

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		
Goto search character (current page)	Description	Example returned values
Focus search box		

[getdate »](#)
[« date timezone set](#)

- [PHP Manual](#)
- [Function Reference](#)
- [Date and Time Related Extensions](#)
- [Date/Time](#)
- [Date/Time Functions](#)

Change language:

[Edit Report a Bug](#)

date

(PHP 4, PHP 5, PHP 7)

date — Format a local time/date

Description ¶

string **date** (string \$format [, int \$timestamp = time()])

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
<i>Day</i>	---	---
<i>d</i>	Day of the month, 2 digits with leading zeros	<i>01 to 31</i>
<i>D</i>	A textual representation of a day, three letters	<i>Mon through Sun</i>
<i>j</i>	Day of the month without leading zeros	<i>1 to 31</i>

Format character (lowercase L)	Description	Example returned values
<i>l</i>	A full textual representation of the week	<i>Saturday</i>
<i>N</i>	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	1 (for Monday) through 7 (for Sunday)
<i>S</i>	English ordinal suffix for the day of the month, 2 characters	<i>st, nd, rd or th.</i> Works well with <i>j</i>
<i>w</i>	Numeric representation of the day of the week	0 (for Sunday) through 6 (for Saturday)
<i>z</i>	The day of the year (starting from 0)	0 through 365
<i>Week</i>	---	---
<i>W</i>	ISO-8601 week number of year, weeks starting on Monday (added in PHP 4.1.0)	Example: 42 (the 42nd week in the year)
<i>Month</i>	---	---
<i>F</i>	A full textual representation of a month, such as January or March	<i>January</i> through <i>December</i>
<i>m</i>	Numeric representation of a month, with leading zeros	01 through 12
<i>M</i>	A short textual representation of a month, three letters	<i>Jan</i> through <i>Dec</i>
<i>n</i>	Numeric representation of a month, without leading zeros	1 through 12
<i>t</i>	Number of days in the given month	28 through 31
<i>Year</i>	---	---
<i>L</i>	Whether it's a leap year	1 if it is a leap year, 0 otherwise.
<i>o</i>	ISO-8601 week-numbering year. This has the same value as <i>Y</i> , except that if the ISO week number (<i>W</i>) belongs to the previous or next year, that year is used instead. (added in PHP 5.1.0)	Examples: 1999 or 2003
<i>Y</i>	A full numeric representation of a year, 4 digits	Examples: 1999 or 2003
<i>y</i>	A two digit representation of a year	Examples: 99 or 03
<i>Time</i>	---	---
<i>a</i>	Lowercase Ante meridiem and Post meridiem	<i>am</i> or <i>pm</i>
<i>A</i>	Uppercase Ante meridiem and Post meridiem	<i>AM</i> or <i>PM</i>
<i>B</i>	Swatch Internet time	000 through 999
<i>g</i>	12-hour format of an hour without leading zeros	1 through 12
<i>G</i>	24-hour format of an hour without leading zeros	0 through 23
<i>h</i>	12-hour format of an hour with leading zeros	01 through 12
<i>H</i>	24-hour format of an hour with leading zeros	00 through 23
<i>i</i>	Minutes with leading zeros	00 to 59
<i>s</i>	Seconds, with leading zeros	00 through 59

format character	Description	Example returned values
	Microseconds (added in PHP 5.2.2). Note that date() will always generate <i>000000</i> since it takes <i>integer</i> parameter, whereas DateTime::format() does support microseconds if DateTime was created with microseconds.	
<i>Timezone</i>	---	---
<i>e</i>	Timezone identifier (added in PHP 5.1.0)	Examples: <i>UTC</i> , <i>GMT</i> , <i>Atlantic/Azores</i>
<i>I</i> (capital <i>i</i>)	Whether or not the date is in daylight saving time	<i>1</i> if Daylight Saving Time, <i>0</i> otherwise.
<i>O</i>	Difference to Greenwich time (GMT) in hours	Example: <i>+0200</i>
<i>P</i>	Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3)	Example: <i>+02:00</i>
<i>T</i>	Timezone abbreviation	Examples: <i>EST</i> , <i>MDT</i> ...
<i>Z</i>	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	<i>-43200</i> through <i>50400</i>
<i>Full Date/Time</i>	---	---
<i>c</i>	ISO 8601 date (added in PHP 5)	2004-02-12T15:19:21+00:00 Example: <i>Thu, 21 Dec 2000 16:01:07 +0200</i>
<i>r</i>	» RFC 2822 formatted date	
<i>U</i>	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 ¶

Description

Version

Description

- 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.
- 5.1.1 There are useful [constants](#) of standard date/time formats that can be used to specify the format parameter.

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 54 notes

[up](#)
[down](#)

108

[Jimmy ¶](#)

4 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](#)

22

[ivijan dot stefan at gmail dot com ¶](#)

1 year 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](#)

24

[FiraSEO ¶](#)

3 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](#)

5

[Anonymous ¶](#)

9 months 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

[Charlie ¶](#)

11 months 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](#)

7

[matthew dot hotchen at worldfirst dot com ¶](#)

2 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

[adityabhai at gmail dot com ¶](#)

3 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](#)

6

[@PeteWilliams ¶](#)

6 years ago

If you want to use HTML5's <date> tag, the following code will generate the machine-readable value for the 'datetime' attribute:

```
<?php

/**
 * formats the date passed into format required by 'datetime' attribute of <date> tag
 * if no intDate supplied, uses current date.
 * @param intDate integer optional
 * @return string
 */
function getDateTimeValue( $intDate = null ) {

    $strFormat = 'Y-m-d\TH:i:s.uP';
    $strDate = $intDate ? date( $strFormat, $intDate ) : date( $strFormat ) ;

    return $strDate;
}

echo getDateTimeValue();

?>
```

[up](#)
[down](#)

7

[eduardo at digmotor dot com dot br ¶](#)

7 years ago

Thanks to tcasparr at gmail dot com for the great idea (at least for me) ;)
 I changed the code a little to replicate the functionality of date_parse_from_format, once I don't have PHP 5.3.0 yet. This might be useful for someone. Hope you don't mind changing your code tcasparr at gmail dot com.

```
<?php
/*****
 * Simple function to take in a date format and return array of associated
 * formats for each date element
 *
 * @return array
 * @param string $strFormat
 *
 * Example: Y/m/d g:i:s becomes
 * Array
 * (
 *     [year] => Y
 *     [month] => m
 *     [day] => d
 *     [hour] => g
 *     [minute] => i
 *     [second] => s
 * )
 *
 * This function is needed for PHP < 5.3.0
 *****/
```

```

function dateParseFromFormat($stFormat, $stData)
{
    $aDataRet = array();
    $aPieces = split('/:.\ \-]', $stFormat);
    $aDatePart = split('/:.\ \-]', $stData);
    foreach($aPieces as $key=>$chPiece)
    {
        switch ($chPiece)
        {
            case 'd':
            case 'j':
                $aDataRet['day'] = $aDatePart[$key];
                break;

            case 'F':
            case 'M':
            case 'm':
            case 'n':
                $aDataRet['month'] = $aDatePart[$key];
                break;

            case 'o':
            case 'Y':
            case 'y':
                $aDataRet['year'] = $aDatePart[$key];
                break;

            case 'g':
            case 'G':
            case 'h':
            case 'H':
                $aDataRet['hour'] = $aDatePart[$key];
                break;

            case 'i':
                $aDataRet['minute'] = $aDatePart[$key];
                break;

            case 's':
                $aDataRet['second'] = $aDatePart[$key];
                break;
        }
    }

    return $aDataRet;
}
?>

```

Also, if you need to change the format of dates:

```

<?php
function changeDateFormat($stDate,$stFormatFrom,$stFormatTo)
{
    // When PHP 5.3.0 becomes available to me

```

```
// $date = date_parse_from_format($stFormatFrom,$stDate);
// For now I use the function above
$date = dateParseFromFormat($stFormatFrom,$stDate);
return date($stFormatTo,mktime($date['hour'],
                                $date['minute'],
                                $date['second'],
                                $date['month'],
                                $date['day'],
                                $date['year']));
}
```

?>

[up](#)
[down](#)

9

[bakerj417 at gmail dot com ¶](#)

4 years ago

If you are having an issue getting u to work so is everyone else. The solution that I am using which I found on another site(so not taking credit) is to use this:

```
date("Y/m/d H:i:s"). substr((string)microtime(), 1, 6);
```

that will give you:

```
yyyy/mm/dd hh:ii:ss.uuuuuu
```

hope this helps someone in need!

thanks all

[up](#)
[down](#)

6

[ghotinet ¶](#)

5 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](#)

3

[Bas Vijfwinkel ¶](#)

4 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](#)

4

[Tim Connolly ¶](#)

4 years ago

Here's my solution for looking up the month number by name (used when parsing an 'ls'):

```

<?php
    for($m=1;$m<=12;$m++){
        $month=date("M",mktime(0,0,0,$m,1,2000));
        $mon["$month"]=$m;
    }
?>

```

[up](#)
[down](#)

5

[mel dot boyce at gmail dot com ¶](#)

10 years ago

I've been flicking through the comments looking for some succinct date code and have noticed an alarming number of questions and over-burdened examples related to date mathematics. One of the most useful skills you can utilize when performing date math is taking full advantage of the UNIX timestamp. The UNIX timestamp was built for this kind of work.

An example of this relates to a comment made by james at bandit-dot-co-dot-en-zed. James was looking for a way to calculate the number of days which have passed since a certain date. Rather than using mktime() and a loop, James can subtract the current timestamp from the timestamp of the date in question and divide that by the number of seconds in a day:

```

<?php
$days = floor((time() - strtotime("01-Jan-2006"))/86400);
print("$days days have passed.\n");
?>

```

Another usage could find itself in a class submitted by Kyle M Hall which aids in the creation of timestamps from the recent past for use with MySQL. Rather than the looping and fine tuning of a date, Kyle can use the raw UNIX timestamps (this is untested code):

```
<?php
$ago = 14; // days
$timestamp = time() - ($ago * 86400);
?>
```

Hopefully these two examples of "UNIX-style" timestamp usage will help those finding date mathematics more elusive than it should be.

[up](#)
[down](#)

5

[Anonymous ¶](#)

8 years ago

Correct format for a MySQL DATETIME column is

```
<?php $mysqltime = date ("Y-m-d H:i:s", $phptime); ?>
```

[up](#)
[down](#)

2

[Anonymous ¶](#)

2 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](#)

0

[david dot thomas at elliotthomas dot com dot au ¶](#)

1 month 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')),"<br />";
?>
```

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](#)

2

[Just.Kevin ¶](#)

7 years ago

In order to determine if a year is a leap year an earlier poster suggested simply checking to see if the year is a multiple of four:

```
<?php
function is_leapyear_broken($year = 2004) {
return ($year%4)==0;
}
?>
```

While this will work for the majority of years it will not work on years that are multiples of 100 but not multiples of 400 i.e.(2100).

A function not using php's date() function that will also account for this small anomaly in leap years:

```
<?php
function is_leapyear_working($year = 2004) {
    if(((($year%4==0) && ($year%100!=0)) || $year%400==0) {
        return true;
    }
    return false;
}
?>
```

While `is_leapyear_working` will not return true for the few non-leap years divisible by four I couldn't tell you if this is more or less efficient than using php's `date()` as an even earlier poster suggested:

```
<?php
function is_leapyear($year = 2004) {
$isleap = date('L', strtotime("$year-1-1"));
return $isleap;
}
?>
```

[up](#)
[down](#)

0

[geoffrey dot hoffman at gmail dot com ¶](#)

10 months ago

I just wanted to emphasise that the return value of `date()` is a string, even when the result of your date format string is a number, such as "j" -> a number 1 to 31, or 'N' -> a day number 1 for Monday through 7 for Sunday... it's still returned as a string! "1" or "7"

or "31". This is much more obvious on the "zero-padded" results, but it's worth repeating.

If you aren't careful, you can get stuck in a while loop comparing days of the week with something like:

```
<?php
// Evil! Don't use this! This will never return!
while ( date('N', $time ) !== 7 ) {
    $time = $time - 86400;
}
?>
```

... for example. The result of date() should be cast to an int for numeric comparison with the exact equality operator:

```
<?php
// Works!
while ( (int)date('N', $time ) !== 7 ) {
    $time = $time - 86400;
}
?>
```

Wasted an hour today on that silly mistake.

[up](#)

[down](#)

0

[tkachenko ivan at bk dot ru ¶](#)

1 year ago

```
<?php
/**
 * Convert a strftime format to a date format
 *
 * Unsupported strftime formats : %U, %W, %C, %g, %r, %R, %T, %X, %c, %D, %F, %x
 * Unsupported date formats : S, n, t, L, B, G, u, e, I, P, Z, c, r
 *
 * @param string $strftimeFormat a strftime format
 * @return string
 */
function strftimeFormatToDate($strftimeFormat) {

    $caracs = array(
        "%d" => "d",
        "%a" => "D",
        "%e" => "j",
        "%A" => "l",
        "%U" => "N",
        "%W" => "w",
        "%j" => "z",
        "%V" => "W",
        "%B" => "F",
        "%m" => "m",
        "%b" => "M",
        "%G" => "o",
        "%Y" => "Y",
```

```

        "%y" => "y",
        "%P" => "a",
        "%p" => "A",
        "%l" => "g",
        "%I" => "h",
        "%H" => "H",
        "%M" => "i",
        "%S" => "s",
        "%Z" => "O",
        "%Z" => "T",
        "%S" => "U",
    );
    return strstr((string)$strftimeFormat, $caracs);
}

$strftimeFormat = '%Y-%m-%d %H:%M:%S';
$formatDate = strftimeFormatToDate($strftimeFormat); // Y-m-d H:i:s
?>

```

[up](#)
[down](#)
3

[SpikeDaCruz ¶](#)

10 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

[Anonymous ¶](#)**2 years ago**

To quickly convert date("N") to a 0 based index with Sunday being represented as 0, you can run it against modulus 7:

```
<?php
$first_of_month_index = date('N', strtotime('4/1/1990')) % 7;
?>
```

[up](#)[down](#)

-1

[Leopietroni ¶](#)**4 years ago**

This function will add working day to a given timestamp

```
<?php
function addworkinday($timestamp,$daystoadd){

    $dayoftheweek = date("N",$timestamp);
    $sum =$dayoftheweek +$daystoadd;

while ($sum >= 6) {

    $daystoadd=$daystoadd+1;
    $sum=$sum-1;
}
return $timestamp +(60*60*24*$daystoadd);
```

}

?>

[up](#)[down](#)

-1

[Horst Frank ¶](#)**6 months ago**

in some cases you can't or don't like to change the setlocal(), but you want to get a translated date(). This function is an example for a german translation-table.

```
function ddate ($format,$timestamp) {

$trans = array(
    'Monday'    => 'Montag',
    'Tuesday'   => 'Dienstag',
    'Wednesday' => 'Mittwoch',
    'Thursday'  => 'Donnerstag',
    'Friday'    => 'Freitag',
    'Saturday'  => 'Samstag',
    'Sunday'    => 'Sonntag',
    'Mon'       => 'Mo',
    'Tue'       => 'Di',
    'Wed'       => 'Mi',
    'Thu'       => 'Do',
```

```

    'Fri'      => 'Fr',
    'Sat'      => 'Sa',
    'Sun'      => 'So',
    'January'  => 'Januar',
    'February' => 'Februar',
    'March'    => 'M&auml;r',
    'May'      => 'Mai',
    'June'     => 'Juni',
    'July'     => 'Juli',
    'October'  => 'Oktober',
    'December' => 'Dezember',
    'Mar'      => 'M&auml;r',
    'Oct'      => 'Okt',
    'Dec'      => 'Dez',
);

return strstr(date($format,$timestamp),$trans);

```

[up](#)
[down](#)
-1

[gerben at gerbenwijnja dot nl ¶](#)

4 years ago

I use the function below to calculate the Unix timestamp of the start of a week. It includes a boolean flag to request a GMT offset instead of the current locale setting.

<?php

```

function getWeekOffsetTimestamp($year, $week, $useGmt = false) {
    if ($useGmt) {
        // Backup timezone and set to GMT
        $timezoneSettingBackup = date_default_timezone_get();
        date_default_timezone_set("GMT");
    }

    // According to ISO-8601, January 4th is always in week 1
    $halfwayTheWeek = strtotime($year."0104 +" .($week - 1)." weeks");

    // Subtract days to Monday
    $dayOfTheWeek = date("N", $halfwayTheWeek);
    $daysToSubtract = $dayOfTheWeek - 1;

    // Calculate the week's timestamp
    $unixTimestamp = strtotime("-$daysToSubtract day", $halfwayTheWeek);

    if ($useGmt) {
        // Reset timezone to backup
        date_default_timezone_set($timezoneSettingBackup);
    }

    return $unixTimestamp;
}

```

?>

[up](#)

[down](#)

-2

[Edward Rudd ¶](#)

7 years ago

To actually make use of the "u" (microsecond) you need to use the DateTime object and not the date() function.

For example

```
<?php
$t = microtime(true);
$micro = sprintf("%06d",($t - floor($t)) * 1000000);
$d = new DateTime( date('Y-m-d H:i:s.'.$micro,$t) );
```

```
print $d->format("Y-m-d H:i:s.u");
```

?>

[up](#)

[down](#)

-3

[frank at interactinet dot com ¶](#)

4 years ago

If you want to compare this week with the same week last year, here is some code to get you the time at the beginning of the week. You can then add days, hours, etc to get to the day of the week that you want to know about.

```
<?php
$time_passed = (date('N')-1)* 24 * 3600; // time since start of week in days
$startOfWeek = mktime(0,0,0,date('m'),date('d'),date('Y')) - $time_passed;

$lastyear = $startOfWeek - 365*24*3600;

// make sure time used from last year is the same week of the year
$weekdiff = date('W') - date('W',$lastyear);
if($weekdiff != 0)
{
    $lastyear = $lastyear + ($weekdiff*7*24*3600);
}

$lastyear_time_passed = (date('N',$lastyear)-1) * 24 * 3600; // time since start of
week in days

$startOfWeek_lastyear =
mktime(0,0,0,date('m',$lastyear),date('d',$lastyear),date('Y',$lastyear)) -
$lastyear_time_passed;
?>
```

So now you have the unix time for the start of this week (\$startOfWeek), and the start of the same week last year (\$startOfWeek_lastyear).

You can convert back to datetime format easily:

```
<?php
    echo date('Y-m-d H:i:s',$startOfWeek).'\<br>';
    echo date('Y-m-d H:i:s',$startOfWeek_lastyear).'\<br><br>';

    echo date('l F jS, Y',$startOfWeek).'\<br>';
    echo date('l F jS, Y',$startOfWeek_lastyear);
?>
```

[up](#)
[down](#)

-4

[Chris ¶](#)

4 years ago

Use this to convert the local/UTC hour to the UTC/local hour:

```
<?php
for($utc_to_local = array(), $offset = date('Z'), $h = 0; $h < 24; $utc_to_local[] =
date('G', mktime($h++)+$offset));
$local_to_utc = array_flip($utc_to_local);

echo "2 am local is ", $local_to_utc[2], " UTC";
echo "3 pm UTC is ", $utc_to_local[15], " local";
?>
```

This is useful when you need to do many conversions. Lookup tables are faster than calling date() and mktime() multiple times.

[up](#)
[down](#)

-5

[lb at bostontech dot net ¶](#)

6 years ago

Not sure why this got ignored the first time, but this is an even simpler way to check leap year:

```
<?php
function isLeapYear($year)
    { return (((($year%4==0) && ($year%100)) || $year%400==0) ? (true):(false)); }
?>
```

[up](#)
[down](#)

-7

[webmaster1989 at gmail dot com ¶](#)

4 years ago

Sometimes it is very useful to convert a sql timestamp to an also called NTP time. This is often used as time date notation in XML RSS pages. To convert a timestamp to this NTP notation try the following:

```
<?php
    echo date('D, d M Y h:i:s O', strtotime ($timestamp));
?>
```

[up](#)
[down](#)

-6

[m ocx at yahoo dot com ¶](#)

4 years ago

Here is a cool Date class to implement the date function:

```
<?php
/*
 * @author Gchats
 *
 * Date class
 */
class Date
{
    private $shortDateFormat = "F j, Y";
    private $longDateFormat = "F j, Y, g:i a";
    private $timestamp = 0;

    /**
     * Default constructor
     *
     * @param integer $timestamp unix time stamp
     */
    function __construct($timestamp = 0)
    {
        $this->timestamp = $timestamp;
    }

    /**
     * Returns the given timestamp in the constructor
     *
     * @return integer time stamp
     */
    public function getTime()
    {
        return (int) $this->timestamp;
    }

    /**
     * Returns long formatted date of the given timestamp
     *
     * @access public
     * @return string Long formatted date
     */
    public function long()
    {
        if ( $this->timestamp > 0 )
        {
            return date ( $this->longDateFormat , $this->timestamp );
        }
        else
        {
            return "";
        }
    }
}

/*
```

```

* Returns short formatted date of the given timestamp
*
* @access public
* @return string Short formatted date
*/
public function short()
{
    if ( $this->timestamp > 0 )
    {
        return date ( $this->shortDateFormat , $this->timestamp );
    }
    else
    {
        return "";
    }
}

public function __toString()
{
    return $this->timestamp;
}

```

}
?>

[up](#)
[down](#)

-5

[ttt joe 08 ¶](#)

2 years ago

Just FYI, it's more appropriate to say "UTC", not "GMT". GMT was given up in 1972 and UTC is now the proper way. The reason being G stands for Greenwich, which naturally upset some people.

[up](#)
[down](#)

-7

[scott at keenot dot es ¶](#)

3 years ago

If anyone needs a really fast function for converting a datetime string (i.e. as retrieved from a MySQL DATETIME entry) into a human-friendly time output analogous to date(\$format, \$time), here's a useful function.

```

<?php
function fdate($datetimestring = '1970-01-01 00:00:00', $format = 'U') {
    // Create a datetime object, return it formatted
    // If you want to give credit for this somewhere, thanks.
    // You really don't have to though; this is kinda obvious
    $dt = new DateTime($datetimestring);
    return $dt->format($format);
}
?>

```

The main purpose of this is to reduce lines of code and allow inline coding. For example:

```

<?php
/* ... */

```

```
echo "This page was submitted on ".fdate($row['created'], 'F j, Y g:i:s A')." and last
modified ".fdate($row['modified'], 'F j, Y g:i:s A')."<br />\n";
```

```
/* ... */
```

```
?>
```

[up](#)
[down](#)

-12

[Anonymous ¶](#)

4 years ago

To find last sunday for given date

```
<?php
```

```
    $day = '2012-10-04';
    echo 'last sunday : '.date("Y-m-d",strtotime($day." last Sunday "));
```

```
?>
```

output:

```
last sunday : 2012-09-30
```

[up](#)
[down](#)

-8

[JonathanCross.com ¶](#)

8 years ago

```
<?php
```

```
// A demonstration of the new DateTime class for those
// trying to use dates before 1970 or after 2038.
```

```
?>
```

```
<h2>PHP 2038 date bug demo (php version <?php echo phpversion(); ?></h1>
```

```
<div style='float:left;margin-right:3em;'>
```

```
<h3>OLD Buggy date()</h3>
```

```
<?php
```

```
    $format='F j, Y';
    for ( $i = 1900; $i < 2050; $i++) {
        $datep = "$i-01-01";
```

```
?>
```

```
    Trying: <?php echo $datep; ?> = <?php echo date($format, strtotime($datep)); ?><br>
```

```
<?php
```

```
    }
```

```
?></div>
```

```
<div style='float:left;'>
```

```
<h3>NEW DateTime Class (v 5.2+)</h3><?php
```

```
    for ( $i = 1900; $i < 2050; $i++) {
        $datep = "$i-01-01";
```

```
        $date = new DateTime($datep);
```

```
?>
```

```
    Trying: <?php echo $datep; ?> = <?php echo $date->format($format); ?><br>
```

```
<?php
```

```
    }
```

```
?></div>
```

[up](#)
[down](#)

-11

[Anon ¶](#)

4 years ago

I needed to convert a duration timestamp into H:i:s but whenever I did it kept bringing 5 back as 01:00:05 (due to some DST stuff) so I made this function to replace date(). It has no optimisations but hopefully someone might find it useful:

```
<?php
function get_time_string(){
    $time = 3600+(60*32)+(50); // 01:32:50
    $time_string = '';

    $hours = (int)($time/(60*60));
    if(strlen($hours) > 1){
        $time_string = $hours.'::~';
    }else{
        $time_string = '0'.$hours.'::~';
    }

    $minutes = (int)((~time%(60*60))/(60));
    if($minutes >= 1){
        if(strlen($minutes) > 1){
            $time_string .= $minutes.'::~';
        }else{
            $time_string .= '0'.$minutes.'::~';
        }

        $seconds = (~time%(60*60))%(60);
        if(strlen($seconds) > 1){
            $time_string .= $seconds;
        }else{
            $time_string .= '0'.$seconds;
        }
    }else{
        if(strlen($time) > 1){
            $time_string .= '00:~.~time;
        }else{
            $time_string .= '00:0'~.~time;
        }
    }
    return $time_string;
}
?>
```

[up](#)
[down](#)

-12

[nathan ¶](#)

4 years ago

```
<?php
/* the following variables are set to appropriate
   characters recognized by php version 5 that
   will get the date. To display the date, we have
   to use 'echo' or 'print' to send the variable
   data to the browser
*/
```

```
$day=date("l");
$date=date("j");
$suffix=date("S");
$month=date("F");
$year=date("Y");
echo $day . ", " . $month . " " . $date . $suffix . ", " . $year;
?>
```

rudimentary, simple way to do things, but it gets the job done for someone learning more on the subject.

[up](#)

[down](#)

-8

[stokestack at gmail dot com ¶](#)

4 years ago

If you want to find your server's timezone offset from GMT, it seems as though you could just do:

```
date('Z')
```

to get the number of seconds offset. But PHP requires that you call `date_default_timezone_set()`. So if you have to hard-code a timezone, why not simply hard-code a variable that tells you the offset from GMT? If you set the timezone to GMT, the dates in your database will still be in local time, but `time('Z')` will return zero.

To keep your code portable across servers in different timezones, you can do this:

```
date_default_timezone_set(date_default_timezone_get())
```

This keeps PHP from complaining that you haven't called `date_default_timezone_set()`, but makes your code portable. Ridiculous.

[up](#)

[down](#)

-10

[Anonymous ¶](#)

3 years ago

Was trying to compare dates when I noticed that:

```
<?php
```

```
var_dump(date('d.m.Y', null));//string(10) "01.01.1970"
var_dump(date('d.m.Y', ''));//bool(false)
```

```
?>
```

Thought it's worth mentioning. Caused some weird logs to be produced in our system since this does not evaluate to the same.

[up](#)

[down](#)

-7

[david dot leon at gmx dot com ¶](#)

2 years ago

```
<?php
```

```
//date returns microseconds.
```

```
function mdate($format, $microtime = null) {
    $microtime = explode(' ', ($microtime ? $microtime : microtime()));
    if (count($microtime) != 2) return false;
    $microtime[0] = $microtime[0] * 1000000;
    $format = str_replace('u', $microtime[0], $format);
    return date($format, $microtime[1]);
}
```

?>

```
echo mdate('Y-m-d H:i:s.u');
```

```
2014-05-19 12:41:59.202303
```

[up](#)

[down](#)

-14

[jc ¶](#)

8 years ago

date("W") returns the iso8601 week number, while date("Y") returns the `_current_year`. This can lead to odd results. For example today (dec 31, 2007) it returns 1 for the week and of course 2007 for the year. This is not wrong in a strict sense because iso defines this week as the first of 2008 while we still have 2007.

So, if you don't have another way to safely retrieve the year according to the iso8061 week-date - `strftime("%G")` doesn't work on some systems -, you should be careful when working with `date("W")`.

For most cases `strftime("%W")` should be a safe replacement.

[edit: Much easier is to use "o" (lower case 0) instead of "Y"]

[up](#)

[down](#)

-5

[sanket at webvice dot co dot uk ¶](#)

1 year ago

<?

/**

* This function gives you the next working days based on the buffer

*

* @param \$date must be in YYYY-MM-DD format

* @param int \$buffer

* @param string \$holidays - You can pass either an array of holidays in YYYYMMDD format or a URL for a .ics file

* containing holidays this defaults to the UK govt holiday data for England and Wales

* @return string

*/

```
function getWorkingDays($date,$buffer=1,$holidays='') {
```

```
    if ($holidays==='') $holidays = 'https://www.gov.uk/bank-holidays/england-and-wales.ics';
```

```
    if (!is_array($holidays)) {
```

```
        $ch = curl_init($holidays);
```

```
        curl_setopt($ch,CURLOPT_RETURNTRANSFER,true);
```

```
        $ics = curl_exec($ch);
```

```

    curl_close($ch);
    $ics = explode("\n",$ics);
    $ics = preg_grep('/^DTSTART;/',$ics);
    $holidays = preg_replace('/^DTSTART;VALUE=DATE:(\d{4})(\d{2})
(\d{2}).*/s','$1-$2-$3',$ics);
}

    $addDay = 0;
    while ($buffer--) {
        while (true) {
            $addDay++;
            $newDate = date('Y-m-d', strtotime("$date +$addDay Days"));
            $newDayOfWeek = date('w', strtotime($newDate));
            if ( $newDayOfWeek>0 && $newDayOfWeek<6 && !in_array($newDate,$holidays))
break;
        }
    }

    return $newDate;
}

?>

```

[up](#)[down](#)

-9

[jock ¶](#)

2 years ago

As of PHP 5.3.3, `date('c')` will produce a string like this:

```
2014-06-17T16:22:42+02:00
```

Instead `date (DATE_IS08601)` will produce:

```
2014-06-17T16:23:36+0200
```

which lacks the semicolon in the timezone part. Both are ISO8601 compliant anyway, but I found that the latter has better compatibility with other languages like python.

[up](#)[down](#)

-6

[akshayinbox at gmail dot com ¶](#)

1 year ago

When using `date()`, be sure to "double-escape" certain characters, for example, if printing the word "at", double escape "t" otherwise it will be treated as "tab" simply leading to a space being inserted.

Example:

```
<?php echo "Last updated ".date("M j<\s\u\p>S</\s\u\p>, Y \a\\t h:i A"); ?>
```

will output

```
Last updated Aug 23rd 2015 at 2:47 PM
```

(assuming today is Aug 23rd 2015 and it is 2:47 PM)

[up](#)

[down](#)

-15

[lehal2 at hotmail dot com ¶](#)**4 years ago**

here is an example how you can make numeric days of the week from 1 to 7(Monday to Friday)

```
<?php
$currentdate = mktime(0, 0, 0, date("m") , date("d"), date("Y"));
    echo $day_eg1 = date ('N', $currentdate);
    echo $day_eg2 = date("N", $today+1 * 24 * 3600);
echo $day_eg3= date("N", $today+2 * 24 * 3600);
echo $day_eg4 = date("N", $today+3 * 24 * 3600);
echo $day_eg5 = date("N", $today+4 * 24 * 3600);
echo $day_eg6 = date("N", $today+5 * 24 * 3600);
echo $day_eg7 = date("N", $today+6 * 24 * 3600);
?>
```

[up](#)[down](#)

-16

[blinov vyacheslav AT gmail.com ¶](#)**5 years ago**

It was oblivious and discouraging that it dont mentioned in docs. If you will use W to get week number be aware:

first days of year can be in a week of previous year, and week number always has leading zero

```
<?php

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

?>
```

so you can't rely on number of week given from this function inside your program if you want to use it for some logic

[up](#)[down](#)

-14

[chubby at chicks dot com ¶](#)**8 years ago**

```
<?php
/**
 * Checks wether a date is between an interval
 *
 * Usage:
 *
 * // check if today is older than 2008/12/31
 * var_dump(currentDayIsInInterval('2008/12/31'));
 * // check if today is younger than 2008/12/31
 * var_dump(currentDayIsInInterval(null, '2008/12/31'));
 * // check if today is between 2008/12/01 and 2008/12/31
 * var_dump(currentDayIsInInterval('2008/12/01', '2008/12/31'));
 *
 */
```

```
* Will trigger errors if date is in wrong format, notices if $begin > $end
*
* @param string $begin Date string as YYYY/mm/dd
* @param string $end Date string as YYYY/mm/dd
* @return bool
*/
function currentDayIsInInterval($begin = '', $end = '')
{
    $preg_exp = "[0-9][0-9][0-9][0-9]/[0-9][0-9]/[0-9][0-9]";
    $preg_error = 'Wrong parameter passed to function '.__FUNCTION__.' : Invalide date
format. Please use YYYY/mm/dd.';
    $interval_error = 'First parameter in '.__FUNCTION__.' should be smaller than
second.';
    if(empty($begin))
    {
        $begin = 0;
    }
    else
    {
        if(preg_match($preg_exp,$begin))
        {
            $begin = (int)str_replace('/', '', $begin);
        }
        else
        {
            trigger_error($preg_error, E_USER_ERROR);
        }
    }
    if(empty($end))
    {
        $end = 99999999;
    }
    else
    {
        if(preg_match($preg_exp,$end))
        {
            $end = (int)str_replace('/', '', $end);
        }
        else
        {
            trigger_error($preg_error, E_USER_ERROR);
        }
    }
    if($end < $begin)
    {
        trigger_error($interval_error, E_USER_WARNING);
    }
    $time = time();
    $now = (int)(date('Y', $time).date('m', $time).date('j', $time));
    if($now > $end or $now < $begin)
    {
        return false;
    }
    return true;
}
```

```
}
?>
```

[up](#)
[down](#)

-13

[Jacques Marais ¶](#)

2 years ago

If you want to print something like: Tuesday, the 14th of January, 2014

Use this:

```
<?php
echo date("l", strtotime("now")).', the'.date(" jS", strtotime("now")).' of'.date(" F, Y",
strtotime("now"));
?>
```

This is because you cannot use words in the date string. If you use words in the date string it will be seen as a format character

So if you use:

```
<?php
echo date("l, the jS of F, Y", strtotime("now"));
?>
```

It will print something like: Tuesday, 3108Europe/Berlin 14th 2014f January, 2014

[up](#)
[down](#)

-6

[krejci dot info at seznam dot cz ¶](#)

1 year ago

I recommend to use "/" instead of "-" when creating dates:

```
<?
    if( date( 'd/m' ) >= date_create( '01/09' ) ) { }
?>
```

[up](#)
[down](#)

-3

[Al Roker ¶](#)

5 months ago

When using 'U' to return a UNIX time stamp, you may not get what you expect. In the following example, we try to get the current Unix time stamp for a user in a different timezone.

```
<?php
// Doesn't work
$timezone = new \DateTimeZone($userTimeZone);
$date = new \DateTime('@' . time(), $timezone);
$date->setTimezone($timezone);
$now = $date->format('U');
?>
```

\$now will return the same (the server's current) Unix time stamp regardless which timezone

your user is in.

To get the actual Unix time stamp based on a time zone, replace format('U') as in the following example;

```
<?php
// This works
$timezone = new \DateTimeZone($userTimeZone);
$date = new \DateTime('@' . time(), $timezone);
$date->setTimezone($timezone);
$now = $date->getTimestamp() + $date->getOffset();
?>
```

[up](#)
[down](#)

-2

[aalaap at gmail dot com ¶](#)

2 months ago

If you need to programatically get a list of all the supported date format identifiers, you can use this simple function:

```
<?php

function getDateFormat() {
    return str_split('dDjLNSwzWFmMntLoYyaABgGhHisueIOPTZcrU');
}

?>
```

PHP doesn't have a built-in identifier listing function, but it has one for listing timezone names.

[up](#)
[down](#)

-21

[Ryan ¶](#)

2 years ago

That is just too hard anyone have it easier terms for a lad who only has internet for 5 mis a day cause he has to walk his pet peanut

[up](#)
[down](#)

-21

[Manu Manjunath ¶](#)

2 years ago

If you want to use "u" format specifier for micrseconds without changing to DateTime object, you may write a function as below:

```
<?php
/**
 * Quick replacement to date() function to handle the 'u' format specifier (for microseconds)
 * @param string $format Date format string - the same format string you would pass to date()
 * function
 * @param float $timestamp [optional] Unix timestamp with microseconds - Typically output of
 * <b>microtime(true)</b>
 * @return string Formatted string
 */
function date_with_micro($format, $timestamp = null) {
```

```

if (is_null($timestamp) || $timestamp === false) {
    $timestamp = microtime(true);
}
$timestamp_int = (int) floor($timestamp);
$microseconds = (int) round(($timestamp - floor($timestamp)) * 1000000.0, 0);
$format_with_micro = str_replace("u", $microseconds, $format);
return date($format_with_micro, $timestamp_int);
}
?>

```

You can safely replace your `date()` function with `date_with_micro()`.

[up](#)
[down](#)

-44

[matt](#)

4 years ago

`date()` has some strange behavior at extremely high values:

```

<?php
echo "9223372036854775805: " . date("Y-m-d g:i:s a", 9223372036854775805) . "\n";
echo "9223372036854775806: " . date("Y-m-d g:i:s a", 9223372036854775806) . "\n";
echo "9223372036854775807: " . date("Y-m-d g:i:s a", 9223372036854775807) . "
(0x7FFFFFFFFFFFFFFF)\n";
echo "9223372036854775808: " . date("Y-m-d g:i:s a", 9223372036854775808) . "\n";
echo "9223372036854775809: " . date("Y-m-d g:i:s a", 9223372036854775809) . "\n";
echo "9223372036854775810: " . date("Y-m-d g:i:s a", 9223372036854775810) . "\n";
echo "... \n";
echo "9223372036854776832: " . date("Y-m-d g:i:s a", 9223372036854776832) . "\n";
echo "9223372036854776833: " . date("Y-m-d g:i:s a", 9223372036854776833) . "\n";
echo "... \n";
echo "9223372036854778879: " . date("Y-m-d g:i:s a", 9223372036854778879) . "\n";
echo "9223372036854778880: " . date("Y-m-d g:i:s a", 9223372036854778880) . "\n";
echo "... \n";
echo "9223372036854780928: " . date("Y-m-d g:i:s a", 9223372036854780928) . "\n";
echo "9223372036854780929: " . date("Y-m-d g:i:s a", 9223372036854780929) . "\n";
echo "... \n";
echo "9223372036854782975: " . date("Y-m-d g:i:s a", 9223372036854782975) . "\n";
echo "9223372036854782976: " . date("Y-m-d g:i:s a", 9223372036854782976) . "\n";
echo "... \n";
echo "9223372036854785024: " . date("Y-m-d g:i:s a", 9223372036854785024) . "\n";
echo "9223372036854785025: " . date("Y-m-d g:i:s a", 9223372036854785025) . "\n";
echo "... \n";
echo "9223372036854787071: " . date("Y-m-d g:i:s a", 9223372036854787071) . "\n";
echo "9223372036854787072: " . date("Y-m-d g:i:s a", 9223372036854787072) . "\n";
echo "... \n";
echo "9223372036854789120: " . date("Y-m-d g:i:s a", 9223372036854789120) . "\n";
echo "9223372036854789121: " . date("Y-m-d g:i:s a", 9223372036854789121) . "\n";
echo "... \n";
echo "9223372036854791167: " . date("Y-m-d g:i:s a", 9223372036854791167) . "\n";
echo "9223372036854791168: " . date("Y-m-d g:i:s a", 9223372036854791168) . "\n";
echo "... \n";
echo "9223372036854793215: " . date("Y-m-d g:i:s a", 9223372036854793215) . "\n";
echo "9223372036854793216: " . date("Y-m-d g:i:s a", 9223372036854793216) . "\n";
echo "9223372036854793217: " . date("Y-m-d g:i:s a", 9223372036854793217) . "\n";

```

```
echo "9223372036854793218: " . date("Y-m-d g:i:s a", 9223372036854793218) . "\n";
?>
```

Output:

```
9223372036854775805: 292277026596-12-04 10:30:05 am
9223372036854775806: 292277026596-12-04 10:30:06 am
9223372036854775807: 292277026596-12-04 10:30:07 am (0x7FFFFFFFFFFFFFFF)
9223372036854775808: 292277026596-12-04 10:30:08 am
9223372036854775809: 292277026596-12-04 10:30:08 am
9223372036854775810: 292277026596-12-04 10:30:08 am
...
9223372036854778879: 292277026596-12-04 10:30:08 am
9223372036854778880: 292277026596-12-04 11:04:16 am
...
9223372036854778879: 292277026596-12-04 11:04:16 am
9223372036854778880: 292277026596-12-04 11:38:24 am
...
9223372036854780928: 292277026596-12-04 11:38:24 am
9223372036854780929: 292277026596-12-04 12:12:32 pm
...
9223372036854782975: 292277026596-12-04 12:12:32 pm
9223372036854782976: 292277026596-12-04 12:46:40 pm
...
9223372036854785024: 292277026596-12-04 12:46:40 pm
9223372036854785025: 292277026596-12-04 1:20:48 pm
...
9223372036854787071: 292277026596-12-04 1:20:48 pm
9223372036854787072: 292277026596-12-04 1:54:56 pm
...
9223372036854789120: 292277026596-12-04 1:54:56 pm
9223372036854789121: 292277026596-12-04 2:29:04 pm
...
9223372036854791167: 292277026596-12-04 2:29:04 pm
9223372036854791168: 292277026596-12-04 3:03:12 pm
...
9223372036854793215: 292277026596-12-04 3:03:12 pm
9223372036854793216: 292277026596-12-04 3:03:12 pm
9223372036854793217: -292277022657-01-27 8:37:04 am
9223372036854793218: -292277022657-01-27 8:37:04 am

---
```

So, the last reliable unix timecode is 9223372036854775808 (0x1000000000000000). Not that you would probably ever need a date that high.

[+ 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](#)
 - [gmdate](#)
 - [gmmktime](#)
 - [gmstrftime](#)
 - [idate](#)
 - [localtime](#)
 - [microtime](#)
 - [mktime](#)
 - [strftime](#)
 - [strptime](#)
 - [strtotime](#)
 - [time](#)
 - [timezone abbreviations list](#)
 - [timezone identifiers list](#)
 - [timezone location get](#)
 - [timezone name from abbr](#)
 - [timezone name get](#)
 - [timezone offset get](#)
 - [timezone open](#)
 - [timezone transitions get](#)
 - [timezone version get](#)
- [Copyright © 2001-2016 The PHP Group](#)
 - [My PHP.net](#)
 - [Contact](#)
 - [Other PHP.net sites](#)
 - [Mirror sites](#)
 - [Privacy policy](#)

