DateTime::diff

DateTimeImmutable::diff

DateTimeInterface::diff

date_diff

(PHP 5 >= 5.3.0, PHP 7)

DateTime::diff -- DateTimeImmutable::diff -- DateTimeInterface::diff -- date_diff — Returns the difference between two DateTime objects

Description ¶

Object oriented style

public DateInterval **DateTime::diff** (DateTimeInterface `$datetime2` [, bool `$absolute` = `FALSE`])
public DateInterval **DateTimeImmutable::diff** (DateTimeInterface `$datetime2` [, bool `$absolute` = `FALSE`])
public DateInterval **DateTimeInterface::diff** (DateTimeInterface `$datetime2` [, bool `$absolute` = `FALSE`])

Parameters ¶

`datetime`
The date to compare to.

`absolute`
Should the interval be forced to be positive?

Return Values ¶

The DateInterval object representing the difference between the two dates or `FALSE` on failure.

Examples ¶

Example #1 DateTime::diff() example

Object oriented style

```
<?php
$datetime1 = new DateTime('2009-10-11');
$datetime2 = new DateTime('2009-10-13');
$interval = $datetime1->diff($datetime2);
echo $interval->format('%R%a days');
?>
```

Procedural style

```
<?php
$datetime1 = date_create('2009-10-11');
$datetime2 = date_create('2009-10-13');
$interval = date_diff($datetime1, $datetime2);
echo $interval->format('%R%a days');
?>
```

The above examples will output:

```
+2 days
```

Example #2 DateTime object comparison

> **Note**:
>
> As of PHP 5.2.2, DateTime objects can be compared using comparison operators.

```
<?php
$date1 = new DateTime("now");
$date2 = new DateTime("tomorrow");
 var_dump($date1 == $date2);
var_dump($date1 < $date2);
var_dump($date1 > $date2);
?>
```

The above example will output:

```
bool(false)
bool(true)
bool(false)
```

See Also ¶

- DateInterval::format() - Formats the interval
- DateTime::add() - Adds an amount of days, months, years, hours, minutes and seconds to a DateTime object
- DateTime::sub() - Subtracts an amount of days, months, years, hours, minutes and seconds from a DateTime object

add a note

User Contributed Notes 27 notes

up
down
121

ianlenmac at gmail dot com ¶

5 years ago

```
 It is worth noting, IMO, and it is implied in the docs but not explicitly
stated, that the object on which diff is called is subtracted from the object
that is passed to diff.
 i.e. $now->diff($tomorrow) is positive.
```

up
down
29

php at jcouture dot net ¶

5 years ago

```
 After wrestling with DateTime::diff for a while it finally dawned on me the
problem was both in the formatting of the input string and the formatting of
the output.
The task was to calculate the duration between two date/times.

### Calculating Duration

1. Make sure you have a valid date variable.  Both of these strings are
valid:

<?php

   $strStart = '2013-06-19 18:25';
   $strEnd   = '06/19/13 21:47';

?>

2. Next convert the string to a date variable
~~~
<?php

   $dteStart = new DateTime($strStart);
   $dteEnd   = new DateTime($strEnd);

?>
~~~

3. Calculate the difference
~~~
<?php

   $dteDiff  = $dteStart->diff($dteEnd);

?>
~~~

4. Format the output
```

```
~~~
<?php

   print $dteDiff->format("%H:%I:%S");

?>
~~~

[Modified by moderator for clarify]
```

up
down
14

nospam at oece dot me ¶
4 years ago

```
Using the identical (===) comparision operator in different but equal objects
will return false
<?php
$c = new DateTime( '2014-04-20' );
$d = new DateTime( '2014-04-20' );
var_dump( $d === $d ); var_dump( $d === $c ); var_dump( $d == $c ); ?>
```

up
down
15

crayonviolent at phpfreaks dot com ¶
4 years ago

```
It seems that while DateTime in general does preserve microseconds,
DateTime::diff doesn't appear to account for it when comparing.
Example:

<?php
$val1 = '2014-03-18 10:34:09.939';
$val2 = '2014-03-18 10:34:09.940';

$datetime1 = new DateTime($val1);
$datetime2 = new DateTime($val2);
echo "<pre>";
var_dump($datetime1->diff($datetime2));

if($datetime1 > $datetime2)
  echo "1 is bigger";
else
  echo "2 is bigger";
?>

The var_dump shows that there is no "u" element, and "2 is bigger" is echoed.

To work around this apparent limitation/oversight, you have to additionally
compare using DateTime::format.
```

```
Example:

<?php
if($datetime1 > $datetime2)
  echo "1 is bigger";
else if ($datetime1->format('u') > $datetime2->format('u'))
  echo "1 is bigger";
else
  echo "2 is bigger";
?>
```

up
down
2

cagatay at devmonks dot net ¶
3 years ago

```
Be careful using:
$date1 = new DateTime('now');
$date2 = new DateTime('tomorrow');

$interval = date_diff($date1, $date2);

echo $interval->format('In %a days');

In some situations, this won't say "in 1 days", but "in 0 days".
I think this is because "now" is the current time, while "tomorrow" is the
current day +1 but at a default time, lets say:

Now: 08:00pm, 01.01.2015
Tomorrow: 00:00am, 02.01.2015

In this case, the difference is not 24 hour, so it will says 0 days.

Better use "today", which should also use a default value like:

Today: 00:00am, 01.01.2015
Tomorrow: 00:00am, 02.01.2015

which now is 24 hour and represents 1 day.

This may sound logical and many will say "of course, this is right", but if
you use it in a naiv way (like I did without thinking), you can come to this
moment and facepalm yourself.

Conclusion: "Now" is "Today", but in a different clock time, but still the
same day!
```

up
down
16

acrion at gmail dot com ¶
8 years ago

```
 If you want to quickly scan through the resulting intervals, you can use the
undocumented properties of DateInterval.
The function below returns a single number of years, months, days, hours,
minutes or seconds between the current date and the provided date.  If the
date occurs in the past (is negative/inverted), it suffixes it with 'ago'.

<?php
function pluralize( $count, $text )
{
    return $count . ( ( $count == 1 ) ? ( " $text" ) : ( " ${text}s" ) );
}

 function ago( $datetime )
{
    $interval = date_create('now')->diff( $datetime );
    $suffix = ( $interval->invert ? ' ago' : '' );
    if ( $v = $interval->y >= 1 ) return pluralize( $interval->y, 'year' ) .
$suffix;
    if ( $v = $interval->m >= 1 ) return pluralize( $interval->m, 'month' )
. $suffix;
    if ( $v = $interval->d >= 1 ) return pluralize( $interval->d, 'day' ) .
$suffix;
    if ( $v = $interval->h >= 1 ) return pluralize( $interval->h, 'hour' ) .
$suffix;
    if ( $v = $interval->i >= 1 ) return pluralize( $interval->i, 'minute' )
. $suffix;
    return pluralize( $interval->s, 'second' ) . $suffix;
}
?>
```

up
down
10

astagl at gmail dot com ¶
7 years ago

```
 I needed to get the exact number of days between 2 dates and was relying on
the this diff function, but found that I was getting a peculiar result with:
<?php
    $today = new DateTime(date('2011-11-09'));
    $appt  = new DateTime(date('2011-12-09'));
    $days_until_appt = $appt->diff($today)->d;
?>

This was returning 0 because it was exactly one month.

I had to end up using :

 <?php
    $days_until_appt = $appt->diff($today)->days;
?>

to get 30.
```

up
down
13

***sgmurphy19* ¶**

6 years ago

```
 Though I found a number of people who ran into the issue of 5.2 and lower not
supporting this function, I was unable to find any solid examples to get
around it. Therefore I hope this can help some others:
<?php
function get_timespan_string($older, $newer) {
  $Y1 = $older->format('Y');
  $Y2 = $newer->format('Y');
  $Y = $Y2 - $Y1;

  $m1 = $older->format('m');
  $m2 = $newer->format('m');
  $m = $m2 - $m1;

  $d1 = $older->format('d');
  $d2 = $newer->format('d');
  $d = $d2 - $d1;

  $H1 = $older->format('H');
  $H2 = $newer->format('H');
  $H = $H2 - $H1;

  $i1 = $older->format('i');
  $i2 = $newer->format('i');
  $i = $i2 - $i1;

  $s1 = $older->format('s');
  $s2 = $newer->format('s');
  $s = $s2 - $s1;

  if($s < 0) {
    $i = $i -1;
    $s = $s + 60;
  }
  if($i < 0) {
    $H = $H - 1;
    $i = $i + 60;
  }
  if($H < 0) {
    $d = $d - 1;
    $H = $H + 24;
  }
  if($d < 0) {
    $m = $m - 1;
    $d = $d + get_days_for_previous_month($m2, $Y2);
  }
  if($m < 0) {
    $Y = $Y - 1;
    $m = $m + 12;
  }
  $timespan_string = create_timespan_string($Y, $m, $d, $H, $i, $s);
  return $timespan_string;
}

function get_days_for_previous_month($current_month, $current_year) {
  $previous_month = $current_month - 1;
  if($current_month == 1) {
    $current_year = $current_year - 1; $previous_month = 12;
  }
  if($previous_month == 11 || $previous_month == 9 || $previous_month == 6 ||
$previous_month == 4) {
    return 30;
  }
  else if($previous_month == 2) {
    if(($current_year % 4) == 0) { return 29;
    }
    else {
      return 28;
    }
  }
  else {
    return 31;
  }
}

function create_timespan_string($Y, $m, $d, $H, $i, $s)
{
  $timespan_string = '';
  $found_first_diff = false;
  if($Y >= 1) {
    $found_first_diff = true;
    $timespan_string .= pluralize($Y, 'year').' ';
  }
  if($m >= 1 || $found_first_diff) {
    $found_first_diff = true;
    $timespan_string .= pluralize($m, 'month').' ';
  }
  if($d >= 1 || $found_first_diff) {
    $found_first_diff = true;
    $timespan_string .= pluralize($d, 'day').' ';
  }
  if($H >= 1 || $found_first_diff) {
    $found_first_diff = true;
    $timespan_string .= pluralize($H, 'hour').' ';
  }
  if($i >= 1 || $found_first_diff) {
    $found_first_diff = true;
    $timespan_string .= pluralize($i, 'minute').' ';
  }
  if($found_first_diff) {
    $timespan_string .= 'and ';
  }
  $timespan_string .= pluralize($s, 'second');
  return $timespan_string;
}

function pluralize( $count, $text )
{
  return $count . ( ( $count == 1 ) ? ( " $text" ) : ( " ${text}s" ) );
}
?>
```

up
down
2

toreskobba at gmail dot com ¶
7 years ago

```
When using datediff make sure your time zone is correct, for me on Windows 7
64 bit it behaved very strange when timezone was wrong (I was comparing now
against time in database and exif metadata in photos). For example:
date_default_timezone_set('Europe/Oslo');
```

up
down
3

schindhelm at gmail dot com ¶
7 years ago

```
I found that DateTime::diff isn't as accurate as I thought. I calculated the
age gap between now and a birthdate from before 1970 (unix epoch). Here's
what I got:
Given today is January 21st, 2011:

<?php
date_default_timezone_set('Europe/Berlin');

$birth = new DateTime('1966-01-21');
$today = new DateTime();
$diff = $birth->diff($today);
echo $diff->format('%y');

$birth = new DateTime('1966-01-23');
$today = new DateTime();
$diff = $birth->diff($today);
echo $diff->format('%y');

$birth = new DateTime('1966-01-24'); $today = new DateTime();
$diff = $birth->diff($today);
echo $diff->format('%y'); ?>

When calculating with the date() function it was more accurate (didn't use
seconds/hours for comparison).
```

```
Note that 3 days may be a lot if you want to create invoices and have to
check against a given age to determine if the customer is chargable for taxes
and so on.

 If someone also found this behaviour I'd like to hear about it - give me a
quick mail at schindhelm (at) gmail (dot) com.
Thanks.
```

up
down
3

Dennis C ¶
8 years ago

```
 For those like me who don't yet have PHP 5.3 installed on their host, here's
a simple alternative to get the number of days between two dates in the
format '2010-3-23' or similar acceptable to strtotime().  You need PHP 5.2.
<?php
function date_diff($date1, $date2) {
    $current = $date1;
    $datetime2 = date_create($date2);
    $count = 0;
    while(date_create($current) < $datetime2){
        $current = gmdate("Y-m-d", strtotime("+1 day", strtotime($current)));
        $count++;
    }
    return $count;
}

 echo (date_diff('2010-3-9', '2011-4-10')." days <br \>");
?>
```

up
down
1

amons dot 360 at gmail dot com ¶
5 years ago

```
 So this function is not available for my server's PHP. I created an
alternative.
Convert the datetime into time-stamps, then subtract normally, then convert
the seconds to whatever you want.

<?

$date1 = new DateTime('now');
$date1->modify("-3 hours");

$date2 = new DateTime('now');

$number1 = (int)$date1->format('U');
$number2 = (int)$date2->format('U');
```

```
echo ($number2 - $number1)/60/60; // will print 3

?>

-Suleiman ALAQEL
```

up
down
1

milespickens+php at gmail dot com ¶
7 years ago

```
 I was looking for a way to output X number of days from a given date and
didn't find exactly what I was looking for.  But I got this working.  I hope
this helps you.
This will output the number of days,months, or years difference between NOW
and a April 1st, 2011.

<?php
    $date1 = new DateTime('2011-04-01');
    $date2 = new DateTime("now");
    $interval = $date1->diff($date2);
    $years = $interval->format('%y');
    $months = $interval->format('%m');
    $days = $interval->format('%d');
    if($years!=0){
        $ago = $years.' year(s) ago';
    }else{
        $ago = ($months == 0 ? $days.' day(s) ago' : $months.' month(s)
ago');
    }
    echo $ago;
?>

If I used today, 2011-05-16 as $date1, I could return all 0's in the format.
For example....

<?php
        $date1 = new DateTime('2011-05-161');
    $date2 = new DateTime("now");
    $interval = $date1->diff($date2);
    $diff = $interval->format('%y-%m-%d');
    echo $diff;  ?>
```

up
down
2

wildpenguin at gmail dot com ¶
7 months ago

```
Be careful when using the difference between 'Now' and a future value.
Example:
```

```
// imagine it is 2018-04-20
$date1 = new DateTime('now');
$date2 = new DateTime(date('Y-m-d'));
$date3 = new DateTime("2018-04-30"); // future
 echo $date1->diff($date3)->days; // 9 days
echo $date2->diff($date3)->days; // 10 days
```

up
down
0

sloanlance+php dot net at gmail dot com ¶
10 months ago

```
 When getting the difference between two DateTime objects with fractions of
seconds, DateTime::diff() works under PHP 7.1. However, under PHP 5.6, the
fraction is truncated. It's not clear whether the truncation happens when
getting the DateTime objects' values, during the calculation, or immediately
before returning the result.
```

up
down
0

Anonymous ¶
1 year ago

```
 <?php
$datetime1 = new DateTime('2016-11-30');
$datetime2 = new DateTime('2017-03-01');
$interval = $datetime1->diff($datetime2);
var_dump($interval->days / 30.4375);
$month = $interval->m + $interval->y * 12;
var_dump($month);
Result:
float(2.9897330595483)
int(2) <-- Where is the third month? February has 28/29 days total.

For
<?php
$datetime1 = new DateTime('2016-11-30');
$datetime2 = new DateTime('2017-03-02');

 Result:
float(3.0225872689938)
int(3)
```

up
down
0

csongor at halmai dot hu ¶
2 years ago

```
Be careful, the behaviour depends on the time zones in a weird way.
<?php
    function printDiff($tz) {
        $d1 = new DateTime("2015-06-01", new DateTimeZone($tz));
        $d2 = new DateTime("2015-07-01", new DateTimeZone($tz));
        $diff = $d1->diff($d2);
        print($diff->format("Year: %Y Month: %M Day: %D"). PHP_EOL);
    }
    printDiff("UTC");
    printDiff("Australia/Melbourne");
?>

The result is different:

Year: 00 Month: 01 Day: 00
Year: 00 Month: 00 Day: 30
```

up
down
0

arunajamal at yahoo dot com ¶

2 years ago

```
Similar to what was mentioned by  ianlenmac at gmail dot com
I think its also worth mentioning to note that date_diff($datetime1,
$datetime2) is equivalent to " subtract $datetime1 from $datetime2 " as
opposed to thinking otherwise because of the arrangement of the arguments
so date_diff($now, $tomorrow) is +ve
```

up
down
0

starsniper at in dot com ¶

4 years ago

```
Another Method to compare dates:
<?php
    $d1 = new DateTime('2014-06-07 20:56:00');
    $d2 = new DateTime('2014-06-09 21:09:00');
    echo 'The DateTimes are: <br /> d1: '.$d1->format("d-M-y, h:i
A").'<br /> d2: '.$d2->format("d-M-y, h:i A");

    $date_diff = $d1->diff($d2);            if (
(int)$date_diff->format('%R%a')  >= 0 ){
        echo '<br /><br />The recent date is: '.$d2->format("d-M-y,
h:i A");
        echo '<br /> The older date is: '.$d1->format("d-M-y, h:i
A");
    }else{
        echo '<br /><br />The recent date is: '.$d1->format("d-M-y,
h:i A");
        echo '<br /> The older date is: '.$d2->format("d-M-y, h:i
A");
    }
?>

Output:
The DateTimes are:
d1: 07-Jun-14, 08:56 PM
d2: 09-Jun-14, 09:09 PM
 The recent date is: 09-Jun-14, 09:09 PM
The older date is: 07-Jun-14, 08:56 PM
```

up
down
0

gusarov at ukr.net ¶
5 years ago

```
 for php<5.3
<?php
$date1 = strtotime('2013-07-03 18:00:00');
$date2 = time();
$subTime = $date1 - $date2;
$y = ($subTime/(60*60*24*365));
$d = ($subTime/(60*60*24))%365;
$h = ($subTime/(60*60))%24;
$m = ($subTime/60)%60;
 echo "Difference between ".date('Y-m-d H:i:s',$date1)." and ".date('Y-m-d
H:i:s',$date2)." is:\n";
echo $y." years\n";
echo $d." days\n";
echo $h." hours\n";
echo $m." minutes\n";
?>
```

up
down
0

Anonymous ¶
9 years ago

```
 You don't need to calculate the exact difference if you just want to know
what date comes earlier:
<?php

date_default_timezone_set('Europe/Madrid');

$d1 = new DateTime('1492-01-01');
$d2 = new DateTime('1492-12-31');

var_dump($d1 < $d2);
var_dump($d1 > $d2);
var_dump($d1 == $d2);
```

```
?>
```

```
bool(true)
bool(false)
bool(false)
```

up
down
-2

radu dot potop at wooptoo dot com ¶
6 years ago

```
Keep in mind that diff will convert the two DateTime objects from local time
to UTC.
```

up
down
-1

devegpat at gmail dot com ¶
3 years ago

```
<?PHP
if($daysbetween > $gap){
  printf("Enter a date within next %d days the difference is
%d",$gap,$daysbetween);
}
else{
  printf("Date is valid and difference  is %d",$daysbetween);
}
?>
```

up
down
-1

Daniel Klein ¶
1 year ago

```
WARNING!!!
Although you CAN directly compare DateTime objects, you will get nonintuitive
results if the other object is not also DateTime compatible.

I just found a subtle bug in my code because it was comparing a date against
an uninitialised variable.

<?php
$now = new DateTime();
$other_date = null;
var_dump($now < $other_date);   var_dump($now > $other_date);   ?>

It's better to use DateTime::diff() because the return value will only be a
DateInterval object if the types were compatible, otherwise it will be false.
```

```
 <?php
$then = new DateTime('yesterday');
$now = new DateTime();
$other_date = null;
var_dump($now->diff($other_date));  var_dump($now->diff($then));    ?>

If a DateInterval object was returned, you can check the 'invert' property to
see if the second date is before the first date or not. DateInterval::invert
will be 1 if the second date is before the first date, and 0 if the the
second date is on or after the first date.
```

up
down
-24

Anonymous ¶
7 years ago

```
 $dateTime = new DateTime('2011-08-01 00:00:00');
echo $dateTime->diff(new DateTime('2011-10-01 00:00:01'))->format('%m');
 will return 1, instead of 2 ...
```

up
down
-8

qrworld.net ¶
3 years ago

```
 Here you have in this post
http://softontherocks.blogspot.com/2014/12/calcular-la-edad-con-php.html the
code to get the age of a person specifying the date of birth:
 function getAge($birthdate){
    $adjust = (date("md") >= date("md", strtotime($birthdate))) ? 0 : -1; //
Si aún no hemos llegado al día y mes en este año restamos 1
    $years = date("Y") - date("Y", strtotime($birthdate)); // Calculamos el
número de años
    return $years + $adjust; // Sumamos la diferencia de años más el ajuste
}
```