Calculate relative time in C#

1435

537

Given a specific `DateTime` value, how do I display relative time, like:

- 2 hours ago
- 3 days ago
- a month ago

c# datetime time datediff relative-time-span

edited Jun 4 '17 at 15:51

community wiki
23 revs, 20 users 24%
Jeff Atwood

77 What if you want to calculate a relative time from now to Future? – Jhonny D. Cano -Leftware- Mar 26 '09 at 20:42

2 moment.js is a very nice date parsing library.. You can consider using that (server side or client side), depending on your needs. just fyi because nobody mentioned it here – code ninja Jun 30 '14 at 13:26

1 There is the .net package github.com/NickStrupat/TimeAgo which pretty much does what is being asked. – Rossco Jun 8 '16 at 11:50

37 Answers

1 2 next

937

Jeff, your code is nice but could be clearer with constants (as suggested in Code Complete).

```
const int SECOND = 1;
```

```
double delta = Math.Abs(ts.TotalSeconds);

if (delta < 1 * MINUTE)
  return ts.Seconds == 1 ? "one second ago" : ts.Seconds + " seconds ago";

if (delta < 2 * MINUTE)
  return "a minute ago";

if (delta < 45 * MINUTE)
  return ts.Minutes + " minutes ago";

if (delta < 90 * MINUTE)
  return "an hour ago";

if (delta < 24 * HOUR)
  return ts.Hours + " hours ago";

if (delta < 48 * HOUR)
  return "yesterday";

if (delta < 30 * DAY)
  return ts.Days + " days ago";

if (delta < 12 * MONTH)
{
  int months = Convert.ToInt32(Math.Floor((double)ts.Days / 30));
  return months <= 1 ? "one month ago" : months + " months ago";
}
else
{
  int years = Convert.ToInt32(Math.Floor((double)ts.Days / 365));
  return years <= 1 ? "one year ago" : years + " years ago";
}
```

edited May 23 '17 at 12:10

community wiki
9 revs, 6 users 78%
Vincent Robert

210 I hate such constants with a passion. Does this look wrong to anyone? `Thread.Sleep(1 * MINUTE)` ? Because it's wrong by a factor of 1000. – Roman Starkov Aug 17 '10 at 17:06

MinutesPerHour = 60; SecondsPerHour = MinutesPerHour * SecondsPerHour; etc. Just calling it MINUTE=60 doesn't allow the reader to determine what the value is. – slolife Aug 29 '12 at 16:21

12 Why nobody (except Joe) care about the wrong 'Yesterday' or 'days ago' value ??? Yesterday is not an hour calculation, but a day to day calculation. So yes, this is a wrong code at least in two frequent case. – CtrlX Sep 26 '13 at 15:47

jquery.timeago plugin

358

Jeff, because Stack Overflow uses jQuery extensively, I recommend the jquery.timeago plugin.

Benefits:

- Avoid timestamps dated "1 minute ago" even though the page was opened 10 minutes ago; timeago refreshes automatically.
- You can take full advantage of page and/or fragment caching in your web applications, because the timestamps aren't calculated on the server.
- You get to use microformats like the cool kids.

Just attach it to your timestamps on DOM ready:

```
jQuery(document).ready(function() {
    jQuery('abbr.timeago').timeago();
});
```

This will turn all `abbr` elements with a class of timeago and an ISO 8601 timestamp in the title:

```
<abbr class="timeago" title="2008-07-17T09:24:17Z">July 17, 2008</abbr>
```

into something like this:

```
<abbr class="timeago" title="July 17, 2008">4 months ago</abbr>
```

9 revs, 4 users 69%
Ryan McGeary

37 Seb, If you have Javascript disabled, then the string you originally put between the abbr tags is displayed. Typically, this is just a date or time in any format you wish. Timeago degrades gracefully. It doesn't get much simpler. – Ryan McGeary Mar 24 '09 at 4:40

23 Ryan, I suggested that SO use timeago a while ago. Jeff's response made me cry, i suggest you sit down: stackoverflow.uservoice.com/pages/1722-general/suggestions/… – Rob Fonseca-Ensor Dec 26 '09 at 7:26

7 Heh, Thanks Rob. That's okay. It's barely noticeable, especially when only one number changes during the transition, though SO pages have a lot of timestamps. I would have thought he would have at least appreciated the benefits of page caching though, even if he chooses to avoid auto-updates. I'm sure Jeff could have provided feedback to improve the plugin too. I take solace knowing sites like arstechnica.com use it. – Ryan McGeary Dec 26 '09 at 13:56

19 @Rob Fonseca-Ensor - now it's making me cry too. How is an update once per minute, to show accurate information, *in any way* related to text blinking once a second? – Daniel Earwicker Apr 20 '10 at 0:23

23 The question is about C#, I fail to see how a jQuery plugin is relevant. – BartoszKP Nov 30 '15 at 12:25

326

Here's how I do it

```
var ts = new TimeSpan(DateTime.UtcNow.Ticks - dt.Ticks);
double delta = Math.Abs(ts.TotalSeconds);

if (delta < 60)
{
  return ts.Seconds == 1 ? "one second ago" : ts.Seconds + " seconds ago";
}
if (delta < 120)
{
  return "a minute ago";
}
if (delta < 2700) // 45 * 60
{
  return ts.Minutes + " minutes ago";
}
if (delta < 5400) // 90 * 60
```

```
if (delta < 172800) // 48 * 60 * 60
{
  return "yesterday";
}
if (delta < 2592000) // 30 * 24 * 60 * 60
{
  return ts.Days + " days ago";
}
if (delta < 31104000) // 12 * 30 * 24 * 60 * 60
{
  int months = Convert.ToInt32(Math.Floor((double)ts.Days / 30));
  return months <= 1 ? "one month ago" : months + " months ago";
}
int years = Convert.ToInt32(Math.Floor((double)ts.Days / 365));
return years <= 1 ? "one year ago" : years + " years ago";
```

Suggestions? Comments? Ways to improve this algorithm?

edited Jan 12 '18 at 16:10

community wiki
12 revs, 6 users 52%
Jeff Atwood

106 "< 48*60*60s" is a rather unconventional definition for "yesterday". If it's 9am on Wednesday, would you really think of 9:01am on Monday as "yesterday". I'd have thought an algorithm for yesterday or "n days ago" should consider before/after midnight. – Joe Feb 1 '09 at 19:33

135 Compilers are usually pretty good at pre-calculating constant expressions, like 24 * 60 * 60, so you can directly use those instead of calculating it yourself to be 86400 and putting the original expression in comments – zvolkov Jun 11 '09 at 12:50

24 noticed that this function excludes weeks – jray Jan 26 '10 at 21:03

11 @bzlm I think I did for a project I was working on. My motivation here was to alert others that weeks were omitted from this code sample. As to how to do that, it seemed pretty straight forward to me. – jray Nov 9 '10 at 23:38

8 I think that good way to improve algorithm is displaying 2 units like "2 month 21 days ago", "1 hour 40 minutes ago" for increasing accuracy. – Evgeny Levin Feb 1 '12 at 13:42

```
public static string RelativeDate(DateTime theDate)
```

```
    thresholds.Add(minute * 2, "a minute ago");
    thresholds.Add(45 * minute, "{0} minutes ago");
    thresholds.Add(120 * minute, "an hour ago");
    thresholds.Add(day, "{0} hours ago");
    thresholds.Add(day * 2, "yesterday");
    thresholds.Add(day * 30, "{0} days ago");
    thresholds.Add(day * 365, "{0} months ago");
    thresholds.Add(long.MaxValue, "{0} years ago");
    long since = (DateTime.Now.Ticks - theDate.Ticks) / 10000000;
    foreach (long threshold in thresholds.Keys)
    {
        if (since < threshold)
        {
            TimeSpan t = new TimeSpan((DateTime.Now.Ticks - theDate.Ticks));
            return string.Format(thresholds[threshold], (t.Days > 365 ? t.Days / 365 :
(t.Days > 0 ? t.Days : (t.Hours > 0 ? t.Hours : (t.Minutes > 0 ? t.Minutes : (t.Seconds
> 0 ? t.Seconds : 0))))).ToString());
        }
    }
    return "";
}
```

I prefer this version for its conciseness, and ability to add in new tick points. This could be encapsulated with a `Latest()` extension to Timespan instead of that long 1 liner, but for the sake of brevity in posting, this will do. **This fixes the an hour ago, 1 hours ago, by providing an hour until 2 hours have elapsed**

edited Feb 22 '18 at 16:30

community wiki
5 revs, 5 users 65%
DevelopingChris

I am getting all sorts of problems using this function, for instance if you mock 'theDate = DateTime.Now.AddMinutes(-40);' I am getting '40 hours ago', but with Michael's refactormycode response, it returns correct at '40 minutes ago' ? – GONeale Dec 15 '08 at 2:17

i think you are missing a zero, try: long since = (DateTime.Now.Ticks - theDate.Ticks) / 10000000; – robnardo Aug 7 '09 at 19:37

8 Hmm, while this code may work it is incorrect and invalid to assume that the order of the keys in the Dictionary will be in a specific order. The Dictionary uses the Object.GetHashCode() which does not return a long but an int!. If you want these to be sorted then you should use a SortedList<long, string>. What is wrong with the thresholds being evaluated in a set of if/else if/.../else ? You get the same number of comparisons. FYI the hash for long.MaxValue turns out to be the same as int.MinValue! – CodeMonkeyKing Nov 9 '11 at 4:47

which threshold you're using) - even if -3 does not create a date in the past year (I have tested this in December, so in this case it should not happen). – Matt Nov 22 '16 at 12:00

71

Here a rewrite from Jeffs Script for PHP:

```php
define("SECOND", 1);
define("MINUTE", 60 * SECOND);
define("HOUR", 60 * MINUTE);
define("DAY", 24 * HOUR);
define("MONTH", 30 * DAY);
function relativeTime($time)
{
    $delta = time() - $time;

    if ($delta < 1 * MINUTE)
    {
        return $delta == 1 ? "one second ago" : $delta . " seconds ago";
    }
    if ($delta < 2 * MINUTE)
    {
      return "a minute ago";
    }
    if ($delta < 45 * MINUTE)
    {
        return floor($delta / MINUTE) . " minutes ago";
    }
    if ($delta < 90 * MINUTE)
    {
      return "an hour ago";
    }
    if ($delta < 24 * HOUR)
    {
      return floor($delta / HOUR) . " hours ago";
    }
    if ($delta < 48 * HOUR)
    {
      return "yesterday";
    }
    if ($delta < 30 * DAY)
```

```
            return $months <= 1 ? "one month ago" : $months . " months ago";
        }
        else
        {
            $years = floor($delta / DAY / 365);
            return $years <= 1 ? "one year ago" : $years . " years ago";
        }
    }
```

edited Feb 6 '12 at 2:04

community wiki
2 revs, 2 users 98%
Thomaschaaf

5 The question is ***C# tagged*** Why ***PHP code*** ? – Kiquenet Mar 6 '17 at 10:31

65

```
public static string ToRelativeDate(DateTime input)
{
    TimeSpan oSpan = DateTime.Now.Subtract(input);
    double TotalMinutes = oSpan.TotalMinutes;
    string Suffix = " ago";

    if (TotalMinutes < 0.0)
    {
        TotalMinutes = Math.Abs(TotalMinutes);
        Suffix = " from now";
    }

    var aValue = new SortedList<double, Func<string>>();
    aValue.Add(0.75, () => "less than a minute");
    aValue.Add(1.5, () => "about a minute");
    aValue.Add(45, () => string.Format("{0} minutes", Math.Round(TotalMinutes)));
    aValue.Add(90, () => "about an hour");
    aValue.Add(1440, () => string.Format("about {0} hours",
Math.Round(Math.Abs(oSpan.TotalHours)))); // 60 * 24
    aValue.Add(2880, () => "a day"); // 60 * 48
    aValue.Add(43200, () => string.Format("{0} days",
Math.Floor(Math.Abs(oSpan.TotalDays)))); // 60 * 24 * 30
```

```
    return aValue.First(n => TotalMinutes < n.Key).Value.Invoke() + Suffix;
}
```

http://refactormycode.com/codes/493-twitter-esque-relative-dates

C# 6 version:

```
static readonly SortedList<double, Func<TimeSpan, string>> offsets =
   new SortedList<double, Func<TimeSpan, string>>
{
    { 0.75, _ => "less than a minute"},
    { 1.5, _ => "about a minute"},
    { 45, x => $"{x.TotalMinutes:F0} minutes"},
    { 90, x => "about an hour"},
    { 1440, x => $"about {x.TotalHours:F0} hours"},
    { 2880, x => "a day"},
    { 43200, x => $"{x.TotalDays:F0} days"},
    { 86400, x => "about a month"},
    { 525600, x => $"{x.TotalDays / 30:F0} months"},
    { 1051200, x => "about a year"},
    { double.MaxValue, x => $"{x.TotalDays / 365:F0} years"}
};

public static string ToRelativeDate(this DateTime input)
{
    TimeSpan x = DateTime.Now - input;
    string Suffix = x.TotalMinutes > 0 ? " ago" : " from now";
    x = new TimeSpan(Math.Abs(x.Ticks));
    return offsets.First(n => x.TotalMinutes < n.Key).Value(x) + Suffix;
}
```

edited May 27 '16 at 19:26

community wiki
5 revs, 4 users 55%
leppie

this is very nice IMO :) This could also be refactored as an extension method? could the dictionary become static so it's only created once and referenced from then after? – Pure.Krome May 6 '09 at 12:29

50

Here's an implementation I added as an extension method to the DateTime class that handles both future and past dates and provides an approximation option that allows you to specify the level of detail you're looking for ("3 hour ago" vs "3 hours, 23 minutes, 12 seconds ago"):

```
using System.Text;

/// <summary>
/// Compares a supplied date to the current date and generates a friendly English
/// comparison ("5 days ago", "5 days from now")
/// </summary>
/// <param name="date">The date to convert</param>
/// <param name="approximate">When off, calculate timespan down to the second.
/// When on, approximate to the largest round unit of time.</param>
/// <returns></returns>
public static string ToRelativeDateString(this DateTime value, bool approximate)
{
    StringBuilder sb = new StringBuilder();

    string suffix = (value > DateTime.Now) ? " from now" : " ago";

    TimeSpan timeSpan = new TimeSpan(Math.Abs(DateTime.Now.Subtract(value).Ticks));

    if (timeSpan.Days > 0)
    {
        sb.AppendFormat("{0} {1}", timeSpan.Days,
          (timeSpan.Days > 1) ? "days" : "day");
        if (approximate) return sb.ToString() + suffix;
    }
    if (timeSpan.Hours > 0)
    {
        sb.AppendFormat("{0}{1} {2}", (sb.Length > 0) ? ", " : string.Empty,
          timeSpan.Hours, (timeSpan.Hours > 1) ? "hours" : "hour");
        if (approximate) return sb.ToString() + suffix;
    }
    if (timeSpan.Minutes > 0)
    {
        sb.AppendFormat("{0}{1} {2}", (sb.Length > 0) ? ", " : string.Empty,
```

```
            if (approximate) return sb.ToString() + suffix;
        }
        if (sb.Length == 0) return "right now";

        sb.Append(suffix);
        return sb.ToString();
    }
```

edited Feb 18 '14 at 14:20

community wiki
6 revs, 3 users 75%
neuracnu

I would recommend computing this on the client side too. Less work for the server.

39

The following is the version that I use (from Zach Leatherman)

```
/*
 * Javascript Humane Dates
 * Copyright (c) 2008 Dean Landolt (deanlandolt.com)
 * Re-write by Zach Leatherman (zachleat.com)
 *
 * Adopted from the John Resig's pretty.js
 * at http://ejohn.org/blog/javascript-pretty-date
 * and henrah's proposed modification
 * at http://ejohn.org/blog/javascript-pretty-date/#comment-297458
 *
 * Licensed under the MIT license.
 */

function humane_date(date_str){
        var time_formats = [
                [60, 'just now'],
                [90, '1 minute'], // 60*1.5
                [3600, 'minutes', 60], // 60*60, 60
                [5400, '1 hour'], // 60*60*1.5
                [86400, 'hours', 3600], // 60*60*24, 60*60
                [129600, '1 day'], // 60*60*24*1.5
                [604800, 'days', 86400], // 60*60*24*7, 60*60*24
```

```
                [4730400000, '1 century'] // 60*60*24*365*100*1.5
        ];

        var time = ('' + date_str).replace(/-/g,"/").replace(/[TZ]/g," "),
                dt = new Date,
                seconds = ((dt - new Date(time) + (dt.getTimezoneOffset() * 60000)) /
1000),
                token = ' ago',
                i = 0,
                format;

        if (seconds < 0) {
                seconds = Math.abs(seconds);
                token = '';
        }

        while (format = time_formats[i++]) {
                if (seconds < format[0]) {
                        if (format.length == 2) {
                                return format[1] + (i > 1 ? token : ''); // Conditional
so we don't return Just Now Ago
                        } else {
                                return Math.round(seconds / format[2]) + ' ' + format[1]
+ (i > 1 ? token : '');
                        }
                }
        }

        // overflow for centuries
        if(seconds > 4730400000)
                return Math.round(seconds / 4730400000) + ' centuries' + token;

        return date_str;
};

if(typeof jQuery != 'undefined') {
        jQuery.fn.humane_dates = function(){
                return this.each(function(){
                        var date = humane_date(this.title);
                        if(date && jQuery(this).text() != date) // don't modify the dom
if we don't have to
                                jQuery(this).text(date);
                });
```

2 revs, 2 users 99%
Jauder Ho

2 The question is ***C# tagged*** Why ***Javascript code*** ? – Kiquenet Mar 6 '17 at 10:37

34

There are also a package called Humanizer on Nuget and it actually works really well

```
DateTime.UtcNow.AddHours(-30).Humanize() => "yesterday"
DateTime.UtcNow.AddHours(-2).Humanize() => "2 hours ago"

DateTime.UtcNow.AddHours(30).Humanize() => "tomorrow"
DateTime.UtcNow.AddHours(2).Humanize() => "2 hours from now"

TimeSpan.FromMilliseconds(1299630020).Humanize() => "2 weeks"
TimeSpan.FromMilliseconds(1299630020).Humanize(3) => "2 weeks, 1 day, 1 hour"
```

Scott Hanselman has a writeup on it on his blog

answered Apr 9 '14 at 11:50 community wiki
Karl-Henrik

1 friendly note: On .net 4.5 or above don't install complete Humanizer... only install Humanizer.Core part of it.. cause other language packages are not supported on this version – Ahmad Mar 10 '17 at 18:50

So useful! This answer must be much much higher in this list. If I had 100 votes, I'd give it to this. Apparently (coming from JS-land), searching for this package was not easy. – kumar_harsh May 27 '17 at 23:54

@jeff

IMHO yours seems a little long. However it does seem a little more robust with support for "yesterday" and "years". But in my experience

```
public static string ToLongString(this TimeSpan time)
{
    string output = String.Empty;

    if (time.Days > 0)
        output += time.Days + " days ";

    if ((time.Days == 0 || time.Days == 1) && time.Hours > 0)
        output += time.Hours + " hr ";

    if (time.Days == 0 && time.Minutes > 0)
        output += time.Minutes + " min ";

    if (output.Length == 0)
        output += time.Seconds + " sec";

    return output.Trim();
}
```

edited Aug 1 '08 at 13:16

community wiki
2 revs
Nick Berardi

24

A couple of years late to the party, but I had a requirement to do this for both past and future dates, so I combined Jeff's and Vincent's into this. It's a ternarytastic extravaganza! :)

```
public static class DateTimeHelper
    {
        private const int SECOND = 1;
        private const int MINUTE = 60 * SECOND;
        private const int HOUR = 60 * MINUTE;
        private const int DAY = 24 * HOUR;
        private const int MONTH = 30 * DAY;

        /// <summary>
        /// Returns a friendly version of the provided DateTime, relative to now. E.g.:
"2 days ago", or "in 6 months".
```

```
            {
                return "Right now!";
            }

            bool isFuture = (DateTime.UtcNow.Ticks < dateTime.Ticks);
            var ts = DateTime.UtcNow.Ticks < dateTime.Ticks ? new
TimeSpan(dateTime.Ticks - DateTime.UtcNow.Ticks) : new TimeSpan(DateTime.UtcNow.Ticks -
dateTime.Ticks);

            double delta = ts.TotalSeconds;

            if (delta < 1 * MINUTE)
            {
                return isFuture ? "in " + (ts.Seconds == 1 ? "one second" : ts.Seconds +
" seconds") : ts.Seconds == 1 ? "one second ago" : ts.Seconds + " seconds ago";
            }
            if (delta < 2 * MINUTE)
            {
                return isFuture ? "in a minute" : "a minute ago";
            }
            if (delta < 45 * MINUTE)
            {
                return isFuture ? "in " + ts.Minutes + " minutes" : ts.Minutes + "
minutes ago";
            }
            if (delta < 90 * MINUTE)
            {
                return isFuture ? "in an hour" : "an hour ago";
            }
            if (delta < 24 * HOUR)
            {
                return isFuture ? "in " + ts.Hours + " hours" : ts.Hours + " hours ago";
            }
            if (delta < 48 * HOUR)
            {
                return isFuture ? "tomorrow" : "yesterday";
            }
            if (delta < 30 * DAY)
            {
                return isFuture ? "in " + ts.Days + " days" : ts.Days + " days ago";
            }
            if (delta < 12 * MONTH)
            {
```

```
                int years = Convert.ToInt32(Math.Floor((double)ts.Days / 365));
                return isFuture ? "in " + (years <= 1 ? "one year" : years + " years") :
years <= 1 ? "one year ago" : years + " years ago";
            }
        }
    }
```

edited May 23 '17 at 12:10

community wiki
3 revs
Town

22

Is there an easy way to do this in Java? The `java.util.Date` class seems rather limited.

Here is my quick and dirty Java solution:

```
import java.util.Date;
import javax.management.timer.Timer;

String getRelativeDate(Date date) {
  long delta = new Date().getTime() - date.getTime();
  if (delta < 1L * Timer.ONE_MINUTE) {
    return toSeconds(delta) == 1 ? "one second ago" : toSeconds(delta) + " seconds ago";
  }
  if (delta < 2L * Timer.ONE_MINUTE) {
    return "a minute ago";
  }
  if (delta < 45L * Timer.ONE_MINUTE) {
    return toMinutes(delta) + " minutes ago";
  }
  if (delta < 90L * Timer.ONE_MINUTE) {
    return "an hour ago";
  }
  if (delta < 24L * Timer.ONE_HOUR) {
    return toHours(delta) + " hours ago";
  }
  if (delta < 48L * Timer.ONE_HOUR) {
    return "yesterday";
  }
```

```
    }
    else {
      long years = toYears(delta);
      return years <= 1 ? "one year ago" : years + " years ago";
    }
  }

  private long toSeconds(long date) {
    return date / 1000L;
  }

  private long toMinutes(long date) {
    return toSeconds(date) / 60L;
  }

  private long toHours(long date) {
    return toMinutes(date) / 60L;
  }

  private long toDays(long date) {
    return toHours(date) / 24L;
  }

  private long toMonths(long date) {
    return toDays(date) / 30L;
  }

  private long toYears(long date) {
    return toMonths(date) / 365L;
  }
```

edited Feb 5 '14 at 9:32

community wiki
3 revs, 2 users 97%
Jo Vermeulen

1 The question is ***C# tagged*** Why ***Java code*** ? – Kiquenet Mar 6 '17 at 10:39

```
    if (delta < 60)
    {
        return delta == 1 ? @"one second ago" : [NSString stringWithFormat:@"%i seconds
ago", delta];
    }
    if (delta < 120)
    {
        return @"a minute ago";
    }
    if (delta < 2700)
    {
        return [NSString stringWithFormat:@"%i minutes ago", delta/60];
    }
    if (delta < 5400)
    {
        return @"an hour ago";
    }
    if (delta < 24 * 3600)
    {
        return [NSString stringWithFormat:@"%i hours ago", delta/3600];
    }
    if (delta < 48 * 3600)
    {
        return @"yesterday";
    }
    if (delta < 30 * 24 * 3600)
    {
        return [NSString stringWithFormat:@"%i days ago", delta/(24*3600)];
    }
    if (delta < 12 * 30 * 24 * 3600)
    {
        int months = delta/(30*24*3600);
        return months <= 1 ? @"one month ago" : [NSString stringWithFormat:@"%i months
ago", months];
    }
    else
    {
        int years = delta/(12*30*24*3600);
        return years <= 1 ? @"one year ago" : [NSString stringWithFormat:@"%i years
ago", years];
    }
}
```

19

Given the world and her husband appear to be posting code samples, here is what I wrote a while ago, based on a couple of these answers.

I had a specific need for this code to be localisable. So I have two classes — `Grammar`, which specifies the localisable terms, and `FuzzyDateExtensions`, which holds a bunch of extension methods. I had no need to deal with future datetimes, so no attempt is made to handle them with this code.

I've left some of the XMLdoc in the source, but removed most (where they'd be obvious) for brevity's sake. I've also not included every class member here:

```
public class Grammar
{
    /// <summary> Gets or sets the term for "just now". </summary>
    public string JustNow { get; set; }
    /// <summary> Gets or sets the term for "X minutes ago". </summary>
    /// <remarks>
    ///     This is a <see cref="String.Format"/> pattern, where <c>{0}</c>
    ///     is the number of minutes.
    /// </remarks>
    public string MinutesAgo { get; set; }
    public string OneHourAgo { get; set; }
    public string HoursAgo { get; set; }
    public string Yesterday { get; set; }
    public string DaysAgo { get; set; }
    public string LastMonth { get; set; }
    public string MonthsAgo { get; set; }
    public string LastYear { get; set; }
    public string YearsAgo { get; set; }
    /// <summary> Gets or sets the term for "ages ago". </summary>
    public string AgesAgo { get; set; }

    /// <summary>
    ///     Gets or sets the threshold beyond which the fuzzy date should be
    ///     considered "ages ago".
    /// </summary>
    public TimeSpan AgesAgoThreshold { get; set; }

    /// <summary>
    ///     Initialises a new <see cref="Grammar"/> instance with the
```

```
    { ... }
}
```

The `FuzzyDateString` class contains:

```
public static class FuzzyDateExtensions
{
    public static string ToFuzzyDateString(this TimeSpan timespan)
    {
        return timespan.ToFuzzyDateString(new Grammar());
    }

    public static string ToFuzzyDateString(this TimeSpan timespan,
        Grammar grammar)
    {
        return GetFuzzyDateString(timespan, grammar);
    }

    public static string ToFuzzyDateString(this DateTime datetime)
    {
        return (DateTime.Now - datetime).ToFuzzyDateString();
    }

    public static string ToFuzzyDateString(this DateTime datetime,
       Grammar grammar)
    {
        return (DateTime.Now - datetime).ToFuzzyDateString(grammar);
    }


    private static string GetFuzzyDateString(TimeSpan timespan,
       Grammar grammar)
    {
        timespan = timespan.Duration();

        if (timespan >= grammar.AgesAgoThreshold)
        {
            return grammar.AgesAgo;
        }

        if (timespan < new TimeSpan(0, 2, 0))    // 2 minutes
```

```
            return String.Format(grammar.MinutesAgo, timespan.Minutes);
        }

        if (timespan < new TimeSpan(1, 55, 0))    // 1 hour 55 minutes
        {
            return grammar.OneHourAgo;
        }

        if (timespan < new TimeSpan(12, 0, 0)    // 12 hours
            && (DateTime.Now - timespan).IsToday())
        {
            return String.Format(grammar.HoursAgo, timespan.RoundedHours());
        }

        if ((DateTime.Now.AddDays(1) - timespan).IsToday())
        {
            return grammar.Yesterday;
        }

        if (timespan < new TimeSpan(32, 0, 0, 0)    // 32 days
            && (DateTime.Now - timespan).IsThisMonth())
        {
            return String.Format(grammar.DaysAgo, timespan.RoundedDays());
        }

        if ((DateTime.Now.AddMonths(1) - timespan).IsThisMonth())
        {
            return grammar.LastMonth;
        }

        if (timespan < new TimeSpan(365, 0, 0, 0, 0)    // 365 days
            && (DateTime.Now - timespan).IsThisYear())
        {
            return String.Format(grammar.MonthsAgo, timespan.RoundedMonths());
        }

        if ((DateTime.Now - timespan).AddYears(1).IsThisYear())
        {
            return grammar.LastYear;
        }

        return String.Format(grammar.YearsAgo, timespan.RoundedYears());
    }
```

```
public static bool IsToday(this DateTime date)
{
    return date.DayOfYear == DateTime.Now.DayOfYear && date.IsThisYear();
}
```

and the rounding methods are like this (I've included `RoundedMonths`, as that's a bit different):

```
public static int RoundedDays(this TimeSpan timespan)
{
    return (timespan.Hours > 12) ? timespan.Days + 1 : timespan.Days;
}

public static int RoundedMonths(this TimeSpan timespan)
{
    DateTime then = DateTime.Now - timespan;

    // Number of partial months elapsed since 1 Jan, AD 1 (DateTime.MinValue)
    int nowMonthYears = DateTime.Now.Year * 12 + DateTime.Now.Month;
    int thenMonthYears = then.Year * 12 + then.Month;

    return nowMonthYears - thenMonthYears;
}
```

I hope people find this useful and/or interesting :o)

edited Feb 22 '18 at 16:34

community wiki
5 revs, 3 users 72%
Owen Blacker

18

using Fluent DateTime

```
var dateTime1 = 2.Hours().Ago();
var dateTime2 = 3.Days().Ago();
var dateTime3 = 1.Months().Ago();
var dateTime4 = 5.Hours().FromNow();
```

4 revs, 3 users 60%
Simon

17

In PHP, I do it this way:

```php
<?php
function timesince($original) {
    // array of time period chunks
    $chunks = array(
        array(60 * 60 * 24 * 365 , 'year'),
        array(60 * 60 * 24 * 30 , 'month'),
        array(60 * 60 * 24 * 7, 'week'),
        array(60 * 60 * 24 , 'day'),
        array(60 * 60 , 'hour'),
        array(60 , 'minute'),
    );

    $today = time(); /* Current unix time  */
    $since = $today - $original;

    if($since > 604800) {
    $print = date("M jS", $original);

    if($since > 31536000) {
        $print .= ", " . date("Y", $original);
    }

    return $print;
}

// $j saves performing the count function each time around the loop
for ($i = 0, $j = count($chunks); $i < $j; $i++) {

    $seconds = $chunks[$i][0];
    $name = $chunks[$i][1];

    // finding the biggest chunk (if the chunk fits, break)
    if (($count = floor($since / $seconds)) != 0) {
        break;
```

```
} ?>
```

edited Feb 6 '12 at 2:05

community wiki
2 revs, 2 users 98%
icco

5 The question is ***C# tagged***. Why this ***PHP code*** ? *IMHO, only applies C# code* – Kiquenet Mar 6 '17 at 10:39

14

I thought I'd give this a shot using classes and polymorphism. I had a previous iteration which used sub-classing which ended up having way too much overhead. I've switched to a more flexible delegate / public property object model which is significantly better. My code is very slightly more accurate, I wish I could come up with a better way to generate "months ago" that didn't seem too over-engineered.

I think I'd still stick with Jeff's if-then cascade because it's less code and it's simpler (it's definitely easier to ensure it'll work as expected).

For the below code *PrintRelativeTime.GetRelativeTimeMessage(TimeSpan ago)* returns the relative time message (e.g. "yesterday").

```
public class RelativeTimeRange : IComparable
{
    public TimeSpan UpperBound { get; set; }

    public delegate string RelativeTimeTextDelegate(TimeSpan timeDelta);

    public RelativeTimeTextDelegate MessageCreator { get; set; }

    public int CompareTo(object obj)
    {
        if (!(obj is RelativeTimeRange))
        {
            return 1;
        }
        // note that this sorts in reverse order to the way you'd expect,
        // this saves having to reverse a list later
        return (obj as RelativeTimeRange).UpperBound.CompareTo(UpperBound);
    }
```

```
{
    timeRanges = new List<RelativeTimeRange>{
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromSeconds(1),
            MessageCreator = (delta) =>
            { return "one second ago"; }
        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromSeconds(60),
            MessageCreator = (delta) =>
            { return delta.Seconds + " seconds ago"; }

        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromMinutes(2),
            MessageCreator = (delta) =>
            { return "one minute ago"; }
        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromMinutes(60),
            MessageCreator = (delta) =>
            { return delta.Minutes + " minutes ago"; }
        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromHours(2),
            MessageCreator = (delta) =>
            { return "one hour ago"; }
        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromHours(24),
            MessageCreator = (delta) =>
            { return delta.Hours + " hours ago"; }
        },
        new RelativeTimeRange
        {
            UpperBound = TimeSpan.FromDays(2),
            MessageCreator = (delta) =>
```

```
                { return delta.Days + " days ago"; }
            },
            new RelativeTimeRange
            {
                UpperBound = DateTime.Now.Subtract(DateTime.Now.AddMonths(-2)),
                MessageCreator = (delta) =>
                { return "one month ago"; }
            },
            new RelativeTimeRange
            {
                UpperBound = DateTime.Now.Subtract(DateTime.Now.AddYears(-1)),
                MessageCreator = (delta) =>
                { return (int)Math.Floor(delta.TotalDays / 30) + " months ago"; }
            },
            new RelativeTimeRange
            {
                UpperBound = DateTime.Now.Subtract(DateTime.Now.AddYears(-2)),
                MessageCreator = (delta) =>
                { return "one year ago"; }
            },
            new RelativeTimeRange
            {
                UpperBound = TimeSpan.MaxValue,
                MessageCreator = (delta) =>
                { return (int)Math.Floor(delta.TotalDays / 365.24D) + " years ago"; }
            }
        };

        timeRanges.Sort();
    }

    public static string GetRelativeTimeMessage(TimeSpan ago)
    {
        RelativeTimeRange postRelativeDateRange = timeRanges[0];

        foreach (var timeRange in timeRanges)
        {
            if (ago.CompareTo(timeRange.UpperBound) <= 0)
            {
                postRelativeDateRange = timeRange;
            }
        }
```

5 revs
Wedge

12

When you know the viewer's time zone, it might be clearer to use calendar days at the day scale. I'm not familiar with the .NET libraries so I don't know how you'd do that in C#, unfortunately.

On consumer sites, you could also be hand-wavier under a minute. "Less than a minute ago" or "just now" could be good enough.

answered Aug 15 '08 at 22:42 community wiki markpasc

12

```
using System;
using System.Collections.Generic;
using System.Linq;

public static class RelativeDateHelper
{
    private static Dictionary<double, Func<double, string>> sm_Dict = null;

    private static Dictionary<double, Func<double, string>> DictionarySetup()
    {
        var dict = new Dictionary<double, Func<double, string>>();
        dict.Add(0.75, (mins) => "less than a minute");
        dict.Add(1.5, (mins) => "about a minute");
        dict.Add(45, (mins) => string.Format("{0} minutes", Math.Round(mins)));
        dict.Add(90, (mins) => "about an hour");
        dict.Add(1440, (mins) => string.Format("about {0} hours",
Math.Round(Math.Abs(mins / 60)))); // 60 * 24
        dict.Add(2880, (mins) => "a day"); // 60 * 48
        dict.Add(43200, (mins) => string.Format("{0} days", Math.Floor(Math.Abs(mins /
1440)))); // 60 * 24 * 30
        dict.Add(86400, (mins) => "about a month"); // 60 * 24 * 60
        dict.Add(525600, (mins) => string.Format("{0} months", Math.Floor(Math.Abs(mins
/ 43200)))); // 60 * 24 * 365
```

```
    public static string ToRelativeDate(this DateTime input)
    {
        TimeSpan oSpan = DateTime.Now.Subtract(input);
        double TotalMinutes = oSpan.TotalMinutes;
        string Suffix = " ago";

        if (TotalMinutes < 0.0)
        {
            TotalMinutes = Math.Abs(TotalMinutes);
            Suffix = " from now";
        }

        if (null == sm_Dict)
            sm_Dict = DictionarySetup();

        return sm_Dict.First(n => TotalMinutes < n.Key).Value.Invoke(TotalMinutes) +
Suffix;
    }
}
```

The same as another answer to this question but as an extension method with a static dictionary.

edited May 23 '17 at 10:31

community wiki
2 revs
Chris Charabaruk

What does the dictionary buy you here? – StriplingWarrior May 26 '11 at 21:46

StriplingWarrior: Ease of reading and modifying compared to a switch statement or a stack of if/else statements. The dictionary being static means that it and the Func<,> objects don't have to be created every time we want to use ToRelativeDate; it's created only once, compared to the one I linked in my answer. – Chris Charabaruk May 31 '11 at 4:42

I see. I was just thinking, since the documentation on `Dictionary` states that "The order in which the items are returned is undefined," (msdn.microsoft.com/en-us/library/xfhwa508.aspx) perhaps that's not the best data structure to use when you don't care about lookup times as much as having things stay in order. – StriplingWarrior May 31 '11 at 16:00

StriplingWarrior: I believe LINQ takes that into account when used with `Dictionary` s. If you're still uncomfortable with it, you can use `SortedDictionary` , but my own experience shows that to be unnecessary. – Chris Charabaruk Jun 1 '11 at 2:44

12

```
long delta = new Date().getTime() - date.getTime();
const int SECOND = 1;
const int MINUTE = 60 * SECOND;
const int HOUR = 60 * MINUTE;
const int DAY = 24 * HOUR;
const int MONTH = 30 * DAY;

if (delta < 0L)
{
  return "not yet";
}
if (delta < 1L * MINUTE)
{
  return ts.Seconds == 1 ? "one second ago" : ts.Seconds + " seconds ago";
}
if (delta < 2L * MINUTE)
{
  return "a minute ago";
}
if (delta < 45L * MINUTE)
{
  return ts.Minutes + " minutes ago";
}
if (delta < 90L * MINUTE)
{
  return "an hour ago";
}
if (delta < 24L * HOUR)
{
  return ts.Hours + " hours ago";
}
if (delta < 48L * HOUR)
{
  return "yesterday";
}
if (delta < 30L * DAY)
{
  return ts.Days + " days ago";
}
if (delta < 12L * MONTH)
{
  int months = Convert.ToInt32(Math.Floor((double)ts.Days / 30));
```

```
    return years <= 1 ? "one year ago" : years + " years ago";
}
```

answered Oct 15 '13 at 9:36

community wiki
Premdeep Mohanty

10

@Jeff

> ```
> var ts = new TimeSpan(DateTime.UtcNow.Ticks - dt.Ticks);
> ```

Doing a subtraction on `DateTime` returns a `TimeSpan` anyway.

So you can just do

```
(DateTime.UtcNow - dt).TotalSeconds
```

I'm also surprised to see the constants multiplied-out by hand and then comments added with the multiplications in. Was that some misguided optimisation?

edited Oct 5 '18 at 2:38

community wiki
4 revs, 4 users 80%
Will Dean

9

Java for client-side gwt usage:

```
import java.util.Date;

public class RelativeDateFormat {
```

```
public static String format(Date date) {

 long delta = new Date().getTime() - date.getTime();
 if (delta < 1L * ONE_MINUTE) {
  return toSeconds(delta) == 1 ? "one second ago" : toSeconds(delta)
    + " seconds ago";
 }
 if (delta < 2L * ONE_MINUTE) {
  return "one minute ago";
 }
 if (delta < 45L * ONE_MINUTE) {
  return toMinutes(delta) + " minutes ago";
 }
 if (delta < 90L * ONE_MINUTE) {
  return "one hour ago";
 }
 if (delta < 24L * ONE_HOUR) {
  return toHours(delta) + " hours ago";
 }
 if (delta < 48L * ONE_HOUR) {
  return "yesterday";
 }
 if (delta < 30L * ONE_DAY) {
  return toDays(delta) + " days ago";
 }
 if (delta < 12L * 4L * ONE_WEEK) {
  long months = toMonths(delta);
  return months <= 1 ? "one month ago" : months + " months ago";
 } else {
  long years = toYears(delta);
  return years <= 1 ? "one year ago" : years + " years ago";
 }
}

private static long toSeconds(long date) {
 return date / 1000L;
}

private static long toMinutes(long date) {
 return toSeconds(date) / 60L;
}

private static long toHours(long date) {
```

```
 private static long toMonths(long date) {
  return toDays(date) / 30L;
 }

 private static long toYears(long date) {
  return toMonths(date) / 365L;
 }

}
```

edited Nov 14 '09 at 18:56

community wiki
2 revs
antony.trupe

The question is ***C# tagged***. Why this ***Java code*** ? *IMHO, only applies C# code* – Kiquenet Mar 6 '17 at 10:40

8

Here's the algorithm stackoverflow uses but rewritten more concisely in perlish pseudocode with a bug fix (no "one hours ago"). The function takes a (positive) number of seconds ago and returns a human-friendly string like "3 hours ago" or "yesterday".

```
agoify($delta)
  local($y, $mo, $d, $h, $m, $s);
  $s = floor($delta);
  if($s<=1)            return "a second ago";
  if($s<60)            return "$s seconds ago";
  $m = floor($s/60);
  if($m==1)            return "a minute ago";
  if($m<45)            return "$m minutes ago";
  $h = floor($m/60);
  if($h==1)            return "an hour ago";
  if($h<24)            return "$h hours ago";
  $d = floor($h/24);
  if($d<2)             return "yesterday";
  if($d<30)            return "$d days ago";
  $mo = floor($d/30);
  if($mo<=1)           return "a month ago";
```

edited Sep 23 '08 at 1:39

community wiki
2 revs
dreeves

8

You can use TimeAgo extension from which looks like the following:

```
public static string TimeAgo(this DateTime dateTime)
{
    string result = string.Empty;
    var timeSpan = DateTime.Now.Subtract(dateTime);

    if (timeSpan <= TimeSpan.FromSeconds(60))
    {
        result = string.Format("{0} seconds ago", timeSpan.Seconds);
    }
    else if (timeSpan <= TimeSpan.FromMinutes(60))
    {
        result = timeSpan.Minutes > 1 ?
            String.Format("about {0} minutes ago", timeSpan.Minutes) :
            "about a minute ago";
    }
    else if (timeSpan <= TimeSpan.FromHours(24))
    {
        result = timeSpan.Hours > 1 ?
            String.Format("about {0} hours ago", timeSpan.Hours) :
            "about an hour ago";
    }
    else if (timeSpan <= TimeSpan.FromDays(30))
    {
        result = timeSpan.Days > 1 ?
            String.Format("about {0} days ago", timeSpan.Days) :
            "yesterday";
    }
    else if (timeSpan <= TimeSpan.FromDays(365))
    {
        result = timeSpan.Days > 30 ?
            String.Format("about {0} months ago", timeSpan.Days / 30) :
            "about a month ago";
```

```
    }

    return result;
}
```

Or use jQuery plugin with Razor extension from Timeago.

edited Jul 13 '16 at 23:23
Ken Graham
119 1 1 6

answered Sep 8 '15 at 14:02
Piotr Stapp
15.1k 4 51 92

8

You can reduce the server-side load by performing this logic client-side. View source on some Digg pages for reference. They have the server emit an epoch time value that gets processed by Javascript. This way you don't need to manage the end user's time zone. The new server-side code would be something like:

```
public string GetRelativeTime(DateTime timeStamp)
{
    return string.Format("<script>printdate({0});</script>", timeStamp.ToFileTimeUtc());
}
```

You could even add a NOSCRIPT block there and just perform a ToString().

edited Sep 27 '18 at 9:13

community wiki
3 revs, 3 users 77%
J

8

This, I got from one of Bill Gates' blog. I need to find it on my browser history and I'll give you the link.

The Javascript code to do the same thing (as requested):

```
function posted(t) {
```

```
    else if (diff < (5400)) { return 'about an hour ago'; }
    else if (diff < (86400)) { return 'about ' + (parseInt(diff / 3600)).toString() + '
hours ago'; }
    else if (diff < (172800)) { return '1 day ago'; }
    else {return (parseInt(diff / 86400)).toString() + ' days ago'; }
}
```

Basically, you work in terms of seconds...

edited Sep 27 '18 at 9:14

community wiki
3 revs, 3 users 92%
Buhake Sindi

6

```
/**
 * {@code date1} has to be earlier than {@code date2}.
 */
public static String relativize(Date date1, Date date2) {
    assert date2.getTime() >= date1.getTime();

    long duration = date2.getTime() - date1.getTime();
    long converted;

    if ((converted = TimeUnit.MILLISECONDS.toDays(duration)) > 0) {
        return String.format("%d %s ago", converted, converted == 1 ? "day" : "days");
    } else if ((converted = TimeUnit.MILLISECONDS.toHours(duration)) > 0) {
        return String.format("%d %s ago", converted, converted == 1 ? "hour" : "hours");
    } else if ((converted = TimeUnit.MILLISECONDS.toMinutes(duration)) > 0) {
        return String.format("%d %s ago", converted, converted == 1 ? "minute" :
"minutes");
    } else if ((converted = TimeUnit.MILLISECONDS.toSeconds(duration)) > 0) {
        return String.format("%d %s ago", converted, converted == 1 ? "second" :
"seconds");
    } else {
        return "just now";
    }
}
```

6

I think there is already a number of answers related to this post, but one can use this which is easy to use just like plugin and also easily readable for programmers. Send your specific date, and get its value in string form:

```
public string RelativeDateTimeCount(DateTime inputDateTime)
{
    string outputDateTime = string.Empty;
    TimeSpan ts = DateTime.Now - inputDateTime;

    if (ts.Days > 7)
    { outputDateTime = inputDateTime.ToString("MMMM d, yyyy"); }

    else if (ts.Days > 0)
    {
        outputDateTime = ts.Days == 1 ? ("about 1 Day ago") : ("about " +
ts.Days.ToString() + " Days ago");
    }
    else if (ts.Hours > 0)
    {
        outputDateTime = ts.Hours == 1 ? ("an hour ago") : (ts.Hours.ToString() + "
hours ago");
    }
    else if (ts.Minutes > 0)
    {
        outputDateTime = ts.Minutes == 1 ? ("1 minute ago") : (ts.Minutes.ToString() + "
minutes ago");
    }
    else outputDateTime = "few seconds ago";

    return outputDateTime;
}
```

edited Sep 27 '18 at 9:14

community wiki
3 revs, 3 users 95%
Prashant Gupta

If you want to have an output like `"2 days, 4 hours and 12 minutes ago"` , you need a timespan:

```
timeDiff.Days
timeDiff.Hours
```

etc...

edited Sep 27 '18 at 10:34

Nalaka526
5,933 15 70 112

answered Sep 8 '15 at 14:02

Bgl86
582 4 18

5

```
var ts = new TimeSpan(DateTime.Now.Ticks - dt.Ticks);
```

edited Sep 27 '18 at 9:15

community wiki
2 revs, 2 users 50%
JoeyFur62

1 2 next

protected by Tim Post ♦ Feb 16 '11 at 13:14

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?