Get difference between 2 dates in JavaScript? [duplicate]

Asked 9 years ago Active 4 months ago Viewed 682k times

464

118

This question already has an answer here:
How do I get the number of days between two dates in JavaScript? *35 answers*

How do I get the difference between 2 dates in full days (I don't want any fractions of a day)

```
var date1 = new Date('7/11/2010');
var date2 = new Date('12/12/2010');
var diffDays = date2.getDate() - date1.getDate();
alert(diffDays)
```

I tried the above but this did not work.

javascript date

edited Jul 23 '17 at 11:54
Brett DeWoody
32.9k 21 94 138

asked Jul 11 '10 at 22:19
chobo2
32k 155 417 705

marked as duplicate by Salman A javascript May 14 '15 at 18:35

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

21 Just a side note: do not create Date objects with these kind of strings as input; it's non-standard and up to the browser how those are parsed. Use strings that can be parsed by Date.parse or, rather, use three numeric values as the arguments to `new Date` , e.g. `new Date(2010, 11, 7);` . – Marcel Korpel Jul 11 '10 at 22:53

2 Side-note: Never trust the system time on client-side! It's wrong quite often, you might break your application. – Sliq Mar 15 '15 at 13:06

Also, try a small function at stackoverflow.com/a/53092438/3787376. – Edward Nov 3 '18 at 21:57

6 Answers

712

[Here is one way]():

```
const date1 = new Date('7/13/2010');
const date2 = new Date('12/15/2010');
const diffTime = Math.abs(date2.getTime() - date1.getTime());
const diffDays = Math.ceil(diffTime / (1000 * 60 * 60 * 24));
console.log(diffDays);
```

Observe that we need to enclose the date in quotes. The rest of the code gets the time difference in milliseconds and then divides to get the number of days. Date expects mm/dd/yyyy format.

edited Apr 1 at 12:45

Arman Yeghiazaryan
2,320 1 9 17

answered Jul 11 '10 at 22:24

TNi
10.2k 2 18 19

9 As volkan answered below, the 1000 * 3600 * 24 is the number of milliseconds per day. As for always returning a positive number, that was a feature :) Typically when one talks about the number of days between two dates, that number is positive. If direction matters, just remove the Math.abs(). – TNi Jul 11 '10 at 22:38

1 Likewise, you get 1 day with your method, because `getDate()` returns the day without regards to the month. Thus, you would get 12 - 11 = 1. – TNi Jul 11 '10 at 22:40

5 date already returned time do not need to call .getTime() – volkan er Jul 11 '10 at 22:56

9 For this to work correctly with daylight saving, Math.round should replace Math.ceil. But I think Shyam's answer is the way to go. – Zemljoradnik Jun 24 '13 at 10:31

8 Won't work with a 23 hour 25 hour day in the calculated span. It would be helpful to consult a detailed treatment of UTC (Universal Coordinated Time) and "civil" time standards before devising a calculation such as this. A day is not a always 86,400 seconds, not even in UTC. However, ECMA standards state that it will not have the 58, 59, 61, or 62 second minutes that occur up to twice a year in UTC. You must not assume that offsets from epoch (00:00:00 hours 1 January 1970) in other languages and operating systems will be the same since some of them have 58, 59, 61, and 62 second minutes. – Jim Mar 8 '15 at 3:13

A more correct solution

654 *... since dates naturally have time-zone information, which can span regions with different day light savings adjustments*

Previous answers to this question don't account for cases where the two dates in question span a daylight saving time (DST) change. The date on which the DST change happens will have a duration in milliseconds which is `!= 1000*60*60*24` , so the typical calculation will fail.

You can work around this by first normalizing the two dates to UTC, and then calculating the difference between those two UTC dates.

Now, the solution can be written as,

```
const _MS_PER_DAY = 1000 * 60 * 60 * 24;

// a and b are javascript Date objects
function dateDiffInDays(a, b) {
  // Discard the time and time-zone information.
  const utc1 = Date.UTC(a.getFullYear(), a.getMonth(), a.getDate());
  const utc2 = Date.UTC(b.getFullYear(), b.getMonth(), b.getDate());

  return Math.floor((utc2 - utc1) / _MS_PER_DAY);
}

// test it
const a = new Date("2017-01-01"),
    b = new Date("2017-07-25"),
    difference = dateDiffInDays(a, b);
```

This works because UTC time never observes DST. See Does UTC observe daylight saving time?

p.s. After discussing some of the comments on this answer, once you've understood the issues with javascript dates that span a DST boundary, there is likely more than just one way to solve it. What I provided above is a simple (and tested) solution. I'd be interested to know if there is a simple arithmetic/math based solution instead of having to instantiate the two new Date objects. That could potentially be faster.

edited Jul 24 '18 at 21:09
Eran Shabi
6,802 6 22 42

answered Mar 8 '13 at 8:43
Shyam Habarakada
11.1k 2 26 44

31 this is the ONLY correct answer, i don't understand why other answer if accepted – obenjiro Apr 5 '13 at 8:26

21 probably because @chobo2 has not come back and looked at this question – Shyam Habarakada Apr 6 '13 at 20:47

3 @Ai_boy: While this is the most logically correct answer, there are no values for which it will fail to match the simpler `Math.round((b - a)/ _MS_PER_DAY, 0)` , so it's difficult to support "the ONLY correct answer". – Scott Sauyet Aug 21 '13 at 0:56

7 This answer is wrong. Javascript *Date* objects **are** UTC. Using local values to generate a UTC date will ignore daylight saving, so if the difference was supposed to be local then treating the dates as UTC will remove daylight saving (so the answer may be wrong over a daylight saving boundary). – RobG Sep 29 '14 at 1:39

10 Using UTC instead of a local time zone will correct the problems caused by 23 hour and 25 hour days in civil time systems. UTC also has 58, 59, 61, and 62 second minutes up to twice a year, so a day in UTC will not always have 86,400 seconds. However, ECMA standards state that JavaScript does not incorporate these adjustments so ignoring them when working purely in JavaScript works. However, other system do not ignore them and you must not assume offsets from the epoch moment (00:00:00 January 1, 1970) will always be the same between programming languages and operating systems. – Jim Mar 8 '15 at 3:03

79

Here is a solution using moment.js:

```
var a = moment('7/11/2010','M/D/YYYY');
var b = moment('12/12/2010','M/D/YYYY');
var diffDays = b.diff(a, 'days');
alert(diffDays);
```

I used your original input values, but you didn't specify the format so I assumed the first value was July 11th. If it was intended to be November 7th, then adjust the format to `D/M/YYYY` instead.

answered Sep 7 '13 at 17:27

Matt Johnson
150k 47 300 424

5 Cheers, I was looking for a momentjs solution – jhhoff02 Sep 6 '16 at 12:54

@Matt Johnson: I have req to find the exact difference, I mean, number of days, Hours, Min and seconds remaining to a future date from this time? How to do that? – Imdad Ali May 31 '18 at 17:59

1 @ImdadAli - See stackoverflow.com/a/41876250/634824. – Matt Johnson May 31 '18 at 20:37

Just note that moment.js is a big dependency – codemirror Mar 1 at 10:33

28

```
var date1 = new Date("7/11/2010");
var date2 = new Date("8/11/2010");
var diffDays = parseInt((date2 - date1) / (1000 * 60 * 60 * 24));
```

```
alert(diffDays )
```

answered Jul 11 '10 at 22:27

volkan er
670 4 6

What is 1000 * 60 * 60 * 24? – chobo2 Jul 11 '10 at 22:32

1 date2 - date1 => milliseconds output (1000 * 60 * 60 * 24) => milisecond to day – volkan er Jul 11 '10 at 22:36

3 Isn't parseInt completely unnecessary? – Christian Jul 11 '10 at 23:02

1 Christian: he wanted an integral number. parseInt is probably the worst way to do it. Math.round is the 'normal' way; it rounds instead of truncating. If truncating was desired, 'or'ing the expression with 0 would suffice: `(date2 - date1) / (1000 * 60 * 60 * 24) | 0` – Dagg Nabbit Jul 12 '10 at 0:05

4 Note that this solution isnt completely accurate as, in some cases i.e. 2015-04-06 minus 2015-04-04 gives an erroneous 1 day, all about parseInt() aproach. – ontananza Mar 24 '15 at 22:54

12

I tried lots of ways, and found that using datepicker was the best, but the date format causes problems with JavaScript....

So here's my answer and can be run out of the box.

```
<input type="text" id="startdate">
<input type="text" id="enddate">
<input type="text" id="days">

<script src="https://code.jquery.com/jquery-1.8.3.js"></script>
<script src="https://code.jquery.com/ui/1.10.0/jquery-ui.js"></script>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/redmond/jquery-
ui.css" />
<script>
$(document).ready(function() {

$( "#startdate,#enddate" ).datepicker({
changeMonth: true,
changeYear: true,
firstDay: 1,
dateFormat: 'dd/mm/yy',
})
```

```
$( "#startdate" ).datepicker({ dateFormat: 'dd-mm-yy' });
$( "#enddate" ).datepicker({ dateFormat: 'dd-mm-yy' });

$('#enddate').change(function() {
var start = $('#startdate').datepicker('getDate');
var end   = $('#enddate').datepicker('getDate');

if (start<end) {
var days   = (end - start)/1000/60/60/24;
$('#days').val(days);
}
else {
alert ("You cant come back before you have been!");
$('#startdate').val("");
$('#enddate').val("");
$('#days').val("");
}
}); //end change function
}); //end ready
</script>
```

a Fiddle can be seen here DEMO

edited May 12 '16 at 14:31

trincot
141k 18 109 147

answered May 1 '14 at 9:19

Mikeys4u
947 12 22

This is a similar problem to what I have the problem with this solution is that if the user clicks on the date2 datepicker first, then it fails. – Alexander May 12 '18 at 6:30

2

This is the code to subtract one date from another. This example converts the dates to objects as the getTime() function won't work unless it's an Date object.

```
var dat1 = document.getElementById('inputDate').value;
            var date1 = new Date(dat1)//converts string to date object
            alert(date1);
            var dat2 = document.getElementById('inputFinishDate').value;
            var date2 = new Date(dat2)
            alert(date2);

            var oneDay = 24 * 60 * 60 * 1000; // hours*minutes*seconds*milliseconds
```

```
var diffDays = Math.abs((date1.getTime() - date2.getTime()) / (oneDay));
alert(diffDays);
```

edited May 12 '15 at 14:57

answered May 12 '15 at 14:37

Pec1983

220 3 8