

# How to format a JavaScript date

Asked 8 years, 11 months ago Active 6 days ago Viewed 3.1m times

How can I format a JavaScript date object to print as 10-Aug-2010 ?

1776

javascript

date

date-format

time-format

edited Apr 4 '18 at 6:29



Sunil Garg

5,039

8

57

86

asked Aug 23 '10 at 23:28



leora

44.4k

301

775

1279



314

149 As usual: beware THE MONTH is ZERO-INDEXED ! So January is zero not one... – [Christophe Roussy](#) Nov 19 '15 at 12:45 ✎

10 Also beware, `myDate.getDay()` doesn't return the day of week, but the **location of the weekday** related to the week. `myDate.getDate()` returns the **current weekday**. – [Jimenemex](#) Aug 18 '17 at 19:47

3 For formatting DateTimes in javascript use the `Intl.DateTimeFormat` object. I describe it in my post: [Post](#). I create an online solution for your answer by `Intl.DateTimeFormat` [Check Online](#) – [Iman Bahrapour](#) Oct 13 '17 at 11:05 ✎

2 You can use `toLocaleDateString` – [onmyway133](#) Nov 5 '18 at 13:50

1 It took javascript so long to get URL as a standardized object, where you can pluck out a query param key, grab the protocol, grab the top level domain, etc. They can make ES6 ES7 but still can't just put a standard date time formatter/parser in 2019? It's as if they're thinking "hmmm yes... who on earth using javascript would have a need to deal with times and dates on a regular basis...." – [ahnbizcad](#) Jan 19 at 4:35 ✎

50 Answers

1

2

next



1069



Attention: There are better answers below. This answer was written in 2010 and newer and better solutions have arrived since. The OP should accept another answer.



```
function formatDate(date) {  
  var monthNames = [  
    "January", "February", "March",  
    "April", "May", "June", "July",  
    "August", "September", "October",  
    "November", "December"  
  ];  
  
  var day = date.getDate();  
  var monthIndex = date.getMonth();  
  var year = date.getFullYear();  
  
  return day + ' ' + monthNames[monthIndex] + ' ' + year;  
}  
  
console.log(formatDate(new Date())); // show current date-time in console
```

[Run code snippet](#)[Hide results](#)[Full page](#)

12 August 2019

10:50:39.220

You can edit the array `monthNames` to use Jan, Feb, Mar, etc..



[edited May 13 '18 at 10:23](#)[answered Aug 23 '10 at 23:35](#)**Marko****49.5k**

25

113

146

**327** Really consider using a library like Moment.js or Date.js instead. This problem has been solved many times over. – [Benjamin Oakes](#) Jan 17 '12 at 19:43

- 201 Why don't they include a function in `Date` object to do this? – [Nayan](#) Jul 24 '14 at 15:04
- 60 One important point is that `getMonth()` method returns a 0 based month index so for example January will return 0 February will return 1, etc... – [Marko](#) Oct 31 '14 at 14:54
- 545 [moment.js](#) 2.9.0 is **11.6k** gzipped, this example is **211 bytes** gzipped. – [mrzmyr](#) Mar 30 '15 at 3:50 
- 24 Should be noted that you should never ever, ever, use `document.write()`. Huge security and performance issues. – [Matt Jensen](#) Sep 9 '15 at 17:27 

## Use `toLocaleDateString()`

1594 The `toLocaleDateString()` method returns a string with a language-sensitive representation of the date portion of the date. The locales and options arguments let applications specify the language whose formatting conventions should be used and allow to customize the behavior of the function.

### The values you can passed in options for different keys:

1. **day:**  
The representation of the day.  
Possible values are "numeric", "2-digit".
2. **weekday:**  
The representation of the weekday.  
Possible values are "narrow", "short", "long".
3. **year:**  
The representation of the year.  
Possible values are "numeric", "2-digit".
4. **month:**  
The representation of the month.  
Possible values are "numeric", "2-digit", "narrow", "short", "long".
5. **hour:**  
The representation of the hour.  
Possible values are "numeric", "2-digit".
6. **minute:** The representation of the minute.  
Possible values are "numeric", "2-digit".

## 7. **second:**

The representation of the second.

Possible values are "numeric", 2-digit".

All these keys are optional. You can change the number of options values based on your requirements, and this will also reflect the presence of each date time term.

Note: If you would only like to configure the content options, but still use the current locale, passing `null` for the first parameter will cause an error. Use `undefined` instead.

## For different languages:

1. **"en-US"**: For English

2. **"hi-IN"**: For Hindi

3. **"ja-JP"**: For Japanese

You can use more language options.

## For example

```
var options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };
var today = new Date();

console.log(today.toLocaleDateString("en-US")); // 9/17/2016
console.log(today.toLocaleDateString("en-US", options)); // Saturday, September 17, 2016
console.log(today.toLocaleDateString("hi-IN", options)); // शनिवार, 17 सितंबर 2016
```

Run code snippet

[Expand snippet](#)

You can also use the `toLocaleString()` method for the same purpose. The only difference is this function provides the time when you don't pass any options.

```
// Example
9/17/2016, 1:21:34 PM
```

## References:

- [toLocaleString\(\)](#).
- [toLocaleDateString\(\)](#).

edited Jan 29 at 11:54



pupeno

110k 102 281 459

answered Dec 1 '15 at 8:11



ajeet kanojia

16k 1 6 7

- 24 Was almost about to use `moment.js` for a simple format. Fortunately did an extra google search and find there is already native API doing this. Saved a external dependency. Awesome! – [LeOn - Han Li](#) Sep 29 '17 at 3:19
- 14 Seems like this answer should be the best "current" answer. Also used the option "hour12: true" to use 12-hour vs 24-hour format. Maybe should be added to your summary list in the answer. – [Doug Knudsen](#) Dec 17 '17 at 17:08
- 1 This is the correct modern answer that formats natively and efficiently you leverage the `Intl.DateTimeFormat` class. – [Dan](#) Jan 5 '18 at 7:18
- 1 NEVERMIND :) just use undefined :) – [carinlynchin](#) Mar 22 '18 at 14:39
- 7 I don't get the upvotes on this answer. It does not solve the problem in the question. (i.e. give me a date which looks like 10-Aug-2010). Using `toLocaleDateString()` that is quite difficult. The `date.format` library seems to be the better solution (at least for Node users) – [larwa1n](#) Jun 24 '18 at 8:40

Use the [date.format library](#):

573

```
var dateFormat = require('dateformat');
var now = new Date();
dateFormat(now, "dddd, mmmm dS, yyyy, h:MM:ss TT");
```

returns:

Saturday, June 9th, 2007, 5:46:21 PM

[dateformat on npm](#)

<http://jsfiddle.net/phZr7/1/>

edited Dec 14 '15 at 15:38



Stephan Muller

21k 11 71 113

answered Aug 23 '10 at 23:35



RobertPitt

46.2k 17 102 144

- 4 this might seem like the longer solution but compressed and used on a site that uses dates a fair bit would be the better solution! – [RobertPitt](#) Aug 25 '10 at 18:33
- 5 This solution is also available as an npm package: [npmjs.com/package/dateformat](https://npmjs.com/package/dateformat) – [David](#) Oct 21 '15 at 15:29
- 15 There are 14 open issues with the above plugin. Even I found one :( – [Amit Kumar Gupta](#) Jul 30 '16 at 16:16
- 5 I get require is not defined – [Hooli](#) Nov 12 '16 at 17:40
- 11 OP asked for JS solution – [Luke Pring](#) Aug 25 '17 at 12:23



If you need to quickly format your date using plain JavaScript, use `getDate` , `getMonth + 1` , `getFullYear` , `getHours` and `getMinutes` :

434



```
var d = new Date();

var datestring = d.getDate() + "-" + (d.getMonth()+1) + "-" + d.getFullYear() + " " +
d.getHours() + ":" + d.getMinutes();

// 16-5-2015 9:50
```

Or, if you need it to be padded with zeros:

```
var datestring = ("0" + d.getDate()).slice(-2) + "-" + ("0" + (d.getMonth()+1)).slice(-2)
+ "-" +
d.getFullYear() + " " + ("0" + d.getHours()).slice(-2) + ":" + ("0" +
d.getMinutes()).slice(-2);

// 16-05-2015 09:50
```

edited Aug 11 '15 at 16:25



[lorem monkey](#)  
3,017 3 28 45

answered May 16 '15 at 7:02



[sebastian.i](#)  
4,467 1 6 7

- 44 The prefix "0".slice(-2) is really nice way to assure padded zeros. Melikes – [Nicholi](#) Aug 25 '15 at 23:23

Can you tell how to get format like Monday, March 23 2018?? – [Sachin HR](#) Aug 27 '18 at 12:01

- 4 you can also pad zeros with `.toString().padStart(2, '0')` – [Benny Jobigan](#) Jan 15 at 10:30

- 1 @DmitryonN, if needed, the year can be padded the same way: `("000" + d.getFullYear()).slice(-4)` – [sebastian.i](#) May 10 at 11:19 ✎

2 @BennyJobigan It should be mentioned that `String.padStart()` is only available from ECMAScript 2017. – JHH May 17 at 11:33



361

Well, what I wanted was to convert today's date to a [MySQL](#) friendly date string like 2012-06-23, and to use that string as a parameter in one of my queries. The simple solution I've found is this:

```
var today = new Date().toISOString().slice(0, 10);
```

Keep in mind that the above solution does **not** take into account your timezone offset.

You might consider using this function instead:

```
function toJSONLocal (date) {  
    var local = new Date(date);  
    local.setMinutes(date.getMinutes() - date.getTimezoneOffset());  
    return local.toJSON().slice(0, 10);  
}
```

This will give you the correct date in case you are executing this code around the start/end of the day.

- Example: <http://jsfiddle.net/simo/sapuhzmm/>
- [Date.toISOString](#)
- [Date.toJSON](#)
- [String.slice](#)

edited May 21 at 7:51

answered Jun 23 '12 at 18:49



simo

9,896 6 36 51

7 You can do `new Date(date + " UTC")` to trick the timezone, and you can eliminate the `setMinutes` line. Man, javascript is dirty – Vajk Hermecz Oct 22 '15 at 22:01

50 not Y10K compatible :( – slang Nov 10 '15 at 2:41

19 Y10K compatible version: `var today = new Date().toISOString().slice(0, -14) ;)` – Alex Shaffer Feb 25 '16 at 13:27

15 Or like this `new Date().toISOString().split('T')[0]` – rofrol Jun 2 '16 at 14:57

2 new Date().toISOString().slice(0, 16).replace('T', ' ') to include time – [Gerrie van Wyk](#) Apr 25 '18 at 19:54

## Custom formatting function:

181 For fixed formats, a simple function make the job. The following example generates the international format YYYY-MM-DD:

```
function dateToYMD(date) {  
  var d = date.getDate();  
  var m = date.getMonth() + 1; //Month from 0 to 11  
  var y = date.getFullYear();  
  return '' + y + '-' + (m <= 9 ? '0' + m : m) + '-' + (d <= 9 ? '0' + d : d);  
}  
  
console.log(dateToYMD(new Date(2017,10,5))); // Nov 5
```

[Run code snippet](#)[Expand snippet](#)

The OP format may be generated like:

```
function dateToYMD(date) {  
  var strArray=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct',  
  'Nov', 'Dec'];  
  var d = date.getDate();  
  var m = strArray[date.getMonth()];  
  var y = date.getFullYear();  
  return '' + (d <= 9 ? '0' + d : d) + '-' + m + '-' + y;  
}  
  
console.log(dateToYMD(new Date(2017,10,5))); // Nov 5
```

[Run code snippet](#)[Expand snippet](#)

Note: It is, however, usually not a good idea to extend the JavaScript standard libraries (e.g. by adding this function to the prototype of Date).

A more advanced function could generate configurable output based on a format parameter.



If to write a formatting function is too long, there are plenty of libraries around which does it. Some other answers already enumerate them. But increasing dependencies also has its counter-part.

## Standard ECMAScript formatting functions:

Since more recent versions of ECMAScript, the `Date` class has some specific formatting functions:

**toDateString:** Implementation dependent, show only the date.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toDateString>

```
new Date().toDateString(); // e.g. "Fri Nov 11 2016"
```

**toISOString:** Show ISO 8601 date and time.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toISOString>

```
new Date().toISOString(); // e.g. "2016-11-21T08:00:00.000Z"
```

**toJSON:** Stringifier for JSON.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toJSON>

```
new Date().toJSON(); // e.g. "2016-11-21T08:00:00.000Z"
```

**toLocaleDateString:** Implementation dependent, a date in locale format.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toLocaleDateString>

```
new Date().toLocaleDateString(); // e.g. "21/11/2016"
```

**toLocaleString:** Implementation dependent, a date&time in locale format.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toLocalestring>

```
new Date().toLocaleString(); // e.g. "21/11/2016, 08:00:00 AM"
```

**toLocaleTimeString:** Implementation dependent, a time in locale format.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toLocaleTimeString>

```
new Date().toLocaleTimeString(); // e.g. "08:00:00 AM"
```

**toString:** Generic toString for Date.

<http://www.ecma-international.org/ecma-262/7.0/index.html#sec-date.prototype.toString>

```
new Date().toString(); // e.g. "Fri Nov 21 2016 08:00:00 GMT+0100 (W. Europe Standard Time)"
```

Note: it is possible to generate custom output out of those formatting >

```
new Date().toISOString().slice(0,10); //return YYYY-MM-DD
```

Examples snippets:

```
console.log("1") "+ new Date().toDateString();  
console.log("2") "+ new Date().toISOString();  
console.log("3") "+ new Date().toJSON();  
console.log("4") "+ new Date().toLocaleDateString();  
console.log("5") "+ new Date().toLocaleString();  
console.log("6") "+ new Date().toLocaleTimeString();
```

```
console.log("7) "+ new Date().toString());
console.log("8) "+ new Date().toISOString().slice(0,10));
```

[Run code snippet](#)
[Expand snippet](#)

edited Nov 30 '17 at 20:22

answered Apr 12 '17 at 9:09



Adrian Maire

7,515 6 24 59

2 Thanks for the last one.. Useful for setting the date value of HTML Date inputs. – [daCoda](#) Jan 30 at 1:08

`new Date().toLocaleDateString()` gives you `mm/dd/yyyy` not `dd/mm/yyyy` please correct that one. – [Rajan Verma](#) Jul 2 at 7:17

@RajanVerma: `toLocaleDateString` provides your locale, which is probably `mm/dd/yyyy` because you are in USA. Here, the locale for date is `dd/mm/yyyy` (that is exactly the point of "locale"). I wrote "e.g." because it is not the specification, but an example of output. – [Adrian Maire](#) Jul 10 at 11:09

If you are **already using jQuery UI** in your project you could do it this way:

169

```
var formatted = $.datepicker.formatDate("M d, yy", new Date("2014-07-08T09:02:21.377"));
```

```
// formatted will be 'Jul 8, 2014'
```

Some datepicker date format options to play with are available [here](#).

edited Aug 21 '14 at 20:44

answered Jul 9 '14 at 13:52




Dmitry Pavlov

18.9k 6 66 77

13 As I said - if jQueryUI is used in project already - why not to re-use the datepicker date formatting function? Hey guys, I don't undersatnd why I'm getting negative voting on my answer? Please explain. – [Dmitry Pavlov](#) Aug 6 '14 at 15:20

6 It might be because someone could include jQuery UI just for the date format function, or it might be because the datepicker is an optional part of the library, but probably it's because hating jQuery is fashionable. – [sennett](#) Aug 21 '14 at 13:53

12 I don't think it is possible to completely avoid all strange decisions that someone could do by mistake or by absense of sense. – [Dmitry Pavlov](#) Aug 21 '14 at 20:43

- 6 @sennett: Hating jQuery is fashionable? So is walking around with your pants halfway down your legs, I suppose... which is pretty much what trying to code without jQuery was like for most of JavaScript's history... – [Michael Scheper](#) Sep 22 '16 at 19:10 
- 5 In any case, this is a helpful and entirely reasonable answer—again, 70% of websites use jQuery. It shouldn't be getting downvoted because of developers' religious beliefs. – [Michael Scheper](#) Oct 30 '16 at 21:13



I think you can just use the **non-standard** Date method `toLocaleFormat(formatString)`

127

**formatString:** A format string in the same format expected by the [strftime\(\)](#) function in C.



```
var today = new Date();
today.toLocaleFormat('%d-%b-%Y'); // 30-Dec-2011
```

### References:

- [toLocaleformat](#)
- [strftime](#)

edited Oct 15 '15 at 8:51



**Slava Fomin II**

11.2k 13 73 142

answered Dec 30 '11 at 5:33



**Sébastien**

1,527 1 9 3

- 150 `toLocaleFormat()` appears to only work in Firefox. Both IE and Chrome are failing for me. – [fitzgeraldsteele](#) Jun 11 '12 at 21:02
- 15 Chrome has `.toLocaleString('en')` method. As it seems new browser supports this [developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...](#) – [apocalypz](#) Jul 9 '14 at 12:44
- 7 Read warning here: [developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/...](#) – [Carson Reinke](#) Feb 9 '15 at 20:09
- 6 this would be the best solution if everyone could f\*in implement it. damn you ie/chrome – [santa](#) Feb 26 '16 at 14:10
- 5 @santa: Indeed. Maybe there was a fair reason not to follow Mozilla's lead on this, but the fact that even ES6 doesn't have a standard function for this shows that it's still a hacker's language, not a developer's language. – [Michael Scheper](#) Sep 22 '16 at 19:16



Plain JavaScript is the best pick for small onetimers.

99

On the other hand, if you need more date stuff, [MomentJS](#) is a great solution.

## For example:

```
moment().format('YYYY-MM-DD HH:m:s');    // now() -> 2015-03-24 14:32:20
moment("20111031", "YYYYMMDD").fromNow(); // 3 years ago
moment("20120620", "YYYYMMDD").fromNow(); // 3 years ago
moment().startOf('day').fromNow();        // 11 hours ago
moment().endOf('day').fromNow();          // in 13 hours
```

edited Mar 4 '17 at 22:17



**Peter Mortensen**

14.3k 19 88 116

answered Dec 24 '14 at 10:15



**Mite Mitreski**

2,932 2 24 37

I think you will probably need more date stuff!! – [morhook](#) Mar 15 '17 at 12:29

1 moment is obsolete, use luxon – [Gerry](#) Apr 11 at 15:35

In modern browsers (\*), you can just do this:

92

```
var today = new Date().toLocaleDateString('en-GB', {
  day : 'numeric',
  month : 'short',
  year : 'numeric'
}).split(' ').join('-');
```

Output if executed today (january 24<sup>th</sup>, 2016):

'24-Jan-2016'

(\*) [According to MDN](#), "modern browsers" means Chrome 24+, Firefox 29+, Internet Explorer 11, Edge 12+, Opera 15+ & Safari [nightly build](#).

edited Jun 8 '17 at 14:55

answered Jan 24 '16 at 21:09



**John Slegers**

30.4k 13 160 137

Is there a way to check if this function is supported and if not, default to a simpler solution? – [James Wierzba](#) Sep 7 '16 at 22:28

@JamesWierzba : You could use [this polyfill!](#) – [John Slegers](#) Sep 12 '16 at 7:56

This isn't even listed on caniuse.com :/ – [Charles Wood](#) Oct 17 '17 at 0:45

You should have a look at [date.js](#). It adds many convenient helpers for working with dates, for example, in your case:

50

```
var date = Date.parse('2010-08-10');
console.log(date.toString('dd-MMM-yyyy'));
```

Getting started: <http://www.datejs.com/2007/11/27/getting-started-with-datejs/>

edited May 18 '15 at 15:53



[Peter Mortensen](#)

14.3k 19 88 116

answered Dec 30 '11 at 8:10



[NiKo](#)

9,122 4 39 54

Thanks. This is a very comprehensive and complete library, with a small footprint. – [mitcheljh](#) Jan 14 '18 at 23:06

I think currently I'm getting a number from Date.parse while let date = new Date(fromString) has more functions. Unfortunately to my surprise toString also seems to just display a default without interpreting the passed argument for formatting it. Using NodeJS 11+ toDateString is a shorter output but doesn't take formatting. All I see is a very convoluted toLocaleDateString – [Master James](#) Dec 4 '18 at 11:25

@Sébastien -- alternative all browser support

35

```
new Date(parseInt(496407600)*1000).toLocaleDateString('de-DE', {
  year: 'numeric',
  month: '2-digit',
  day: '2-digit'
}).replace(/\./g, '/');
```

Documentation: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Date/toLocaleDateString](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toLocaleDateString)

answered Oct 2 '14 at 16:40



[user956584](#)

3,718 2 31 43

5 Instead of doing `.replace()`, you could simply use 'en-GB' as locale. :) – [Roberto14](#) Feb 27 '15 at 12:29

1 This is really nice, e.g. `new Date().toLocaleDateString("en-EN", {month: "short", weekday: "short", day: "2-digit", year: "numeric"})` returns "Wed, Sep 06, 2017" – [Pedi T.](#) Sep 6 '17 at 14:03

There is good enough browser support for this. – [desmati](#) Dec 30 '18 at 13:55

I can get your requested format in one line using no libraries and no Date methods, just regex:

34

```
var d = (new Date()).toString().replace(/\S+\s(\S+)\s(\d+)\s(\d+)\s.*/, '$2-$1-$3');
// date will be formatted as "14-Oct-2015" (pass any date object in place of 'new Date()')
```

Update: As [@RobG](#) pointed out, the output of `Date.prototype.toString()` is implementation-dependent. So, use with caution and modify if necessary for your implementations if you use this solution. In my testing, this works reliably in North America where the major browsers (Chrome, Safari, Firefox and IE) all return the same string format.

edited Sep 10 '16 at 3:28

answered Oct 14 '15 at 17:25



[JD Smith](#)

1,446 13 15

`console.log(new Date().toString().replace(/\S+\s(\S+)\s(\d+)\s(\d+)\s.*/, '$2-$1-$3'));` – [John](#) Apr 30 '16 at 11:44

[@André](#) - I agree. If this were my code, I would most certainly include a comment alongside it that explains the regex and gives an example of the input and corresponding output. – [JD Smith](#) Dec 6 '18 at 21:18

Using an ECMAScript Edition 6 (ES6/ES2015) string template:

28

```
let d = new Date();
let formatted = `${d.getFullYear()}-${d.getMonth() + 1}-${d.getDate()}`;
```

If you need to change the delimiters:

```
const delimiter = '/';
let formatted = [d.getFullYear(), d.getMonth() + 1, d.getDate()].join(delimiter);
```

edited May 7 '18 at 13:31

answered Aug 28 '17 at 12:49



vdegenne

4,083 8 47 77

**Packaged Solution:** [Luxon](#)

19

If you want to use a one solution to fit all, I highly recommend using Luxon (a modernized version of [Moment.js](#)) which also does formatting in many locales/languages and tons of other features.



Luxon is hosted on the Moment.js website and developed by a Moment.js developer because Moment.js has limitations that the developer wanted to address but couldn't.

To install:

```
npm install luxon OR yarn add luxon (visit link for other installation methods)
```

Example:

```
luxon.DateTime.fromISO('2010-08-10').toFormat('yyyy-LLL-dd');
```

Yields:

10-Aug-2010

**Manual Solution**

Using similar formatting as Moment.js, [Class DateTimeFormatter \(Java\)](#), and [Class SimpleDateFormat \(Java\)](#), I implemented a comprehensive solution `formatDate(date, patternStr)` where the code is easy to read and modify. You can display date, time, AM/PM, etc. See code for more examples.

Example:

```
formatDate(new Date(), 'EEEE, MMMM d, yyyy HH:mm:ss:S')
```

( `formatDate` is implemented in the code snippet below)

Yields:



Friday, October 12, 2018 18:11:23:445

Try the code out by clicking "Run code snippet."

## Date and Time Patterns

yy = 2-digit year; yyyy = full year

M = digit month; MM = 2-digit month; MMM = short month name; MMMM = full month name

EEEE = full weekday name; EEE = short weekday name

d = digit day; dd = 2-digit day

h = hours am/pm; hh = 2-digit hours am/pm; H = hours; HH = 2-digit hours

m = minutes; mm = 2-digit minutes; aa = AM/PM

s = seconds; ss = 2-digit seconds

S = milliseconds

```
var monthNames = [
  "January", "February", "March", "April", "May", "June", "July",
  "August", "September", "October", "November", "December"
];
var dayOfWeekNames = [
  "Sunday", "Monday", "Tuesday",
  "Wednesday", "Thursday", "Friday", "Saturday"
];
function formatDate(date, patternStr){
  if (!patternStr) {
    patternStr = 'M/d/yyyy';
  }
  var day = date.getDate(),
      month = date.getMonth(),
      year = date.getFullYear(),
      hour = date.getHours(),
      minute = date.getMinutes(),
      second = date.getSeconds(),
      milliseconds = date.getMilliseconds(),
      h = hour % 12,
      hh = twoDigitPad(h),
```

```

    HH = twoDigitPad(hour),
    mm = twoDigitPad(minute),
    ss = twoDigitPad(second),
    aaa = hour < 12 ? 'AM' : 'PM',
    EEEE = dayOfWeekNames[date.getDay()],
    EEE = EEEE.substr(0, 3),
    dd = twoDigitPad(day),
    M = month + 1,
    MM = twoDigitPad(M),
    MMMM = monthNames[month],
    MMM = MMMM.substr(0, 3),
    yyyy = year + "",
    yy = yyyy.substr(2, 2)
;
// checks to see if month name will be used
patternStr = patternStr
    .replace('hh', hh).replace('h', h)
    .replace('HH', HH).replace('H', hour)
    .replace('mm', mm).replace('m', minute)
    .replace('ss', ss).replace('s', second)
    .replace('S', milliseconds)
    .replace('dd', dd).replace('d', day)

    .replace('EEEE', EEEE).replace('EEE', EEE)
    .replace('yyyy', yyyy)
    .replace('yy', yy)
    .replace('aaa', aaa);
if (patternStr.indexOf('MMM') > -1) {
    patternStr = patternStr
        .replace('MMMM', MMMM)
        .replace('MMM', MMM);
}
else {
    patternStr = patternStr
        .replace('MM', MM)
        .replace('M', M);
}
return patternStr;
}
function twoDigitPad(num) {
    return num < 10 ? "0" + num : num;
}
console.log(formatDate(new Date()));
console.log(formatDate(new Date(), 'dd-MMM-yyyy')); //OP's request
console.log(formatDate(new Date(), 'EEEE, MMMM d, yyyy HH:mm:ss.S aaa'));
console.log(formatDate(new Date(), 'EEE, MMM d, yyyy HH:mm'));
console.log(formatDate(new Date(), 'yyyy-MM-dd HH:mm:ss.S'));
console.log(formatDate(new Date(), 'M/dd/yyyy h:mmmaa'));

```

Run code snippet

[Expand snippet](#)

Thank you @Gerry for bringing up Luxon.

edited Aug 6 at 1:22

answered Oct 13 '18 at 4:19



lewdev

3,247 1 15 16

1 By the way, the troublesome `SimpleDateFormat` class was supplanted years ago by the `java.time.format.DateTimeFormatter` class. – [Basil Bourque](#) Oct 13 '18 at 16:40

1 @BasilBourque, noted. They both use the same patterns. I was on a pre-Java8 project for 5 years so I never got exposed to the newer stuff. Thanks! – [lewdev](#) Oct 15 '18 at 20:34

See [ThreeTen-Backport](#) project for Java 6 & 7, to get most of the *java.time* functionality with nearly identical API. – [Basil Bourque](#) Oct 15 '18 at 20:37

@BasilBourque thanks for the reference, but I don't work on that project anymore but I'll definitely keep this in mind when it comes up. – [lewdev](#) Oct 15 '18 at 21:07

Keep in mind that here on Stack Overflow I am speaking to the two million readers of this page, not really you individually. ;-) – [Basil Bourque](#) Oct 15 '18 at 21:09

18 Here's is some code I just wrote to handle the date formatting for a project I'm working on. It mimics the PHP date formatting functionality to suit my needs. Feel free to use it, it's just extending the already existing `Date()` object. This may not be the most elegant solution but it's working for my needs.

```
var d = new Date();
d_string = d.format("m/d/Y h:i:s");

/*****
 * Date class extension
 *
 */
// Provide month names
Date.prototype.getMonthName = function(){
    var month_names = [
        'January',
        'February',
        'March',
        'April',
```

```
        'May',
        'June',
        'July',
        'August',
        'September',
        'October',
        'November',
        'December'
    ];

    return month_names[this.getMonth()];
}

// Provide month abbreviation
Date.prototype.getMonthAbbr = function(){
    var month_abbrs = [
        'Jan',
        'Feb',
        'Mar',
        'Apr',
        'May',
        'Jun',
        'Jul',
        'Aug',
        'Sep',
        'Oct',
        'Nov',
        'Dec'
    ];

    return month_abbrs[this.getMonth()];
}

// Provide full day of week name
Date.prototype.getDayFull = function(){
    var days_full = [
        'Sunday',
        'Monday',
        'Tuesday',
        'Wednesday',
        'Thursday',
        'Friday',
        'Saturday'
    ];

    return days_full[this.getDay()];
};

// Provide full day of week name
Date.prototype.getDayAbbr = function(){
```

```

var days_abbr = [
    'Sun',
    'Mon',
    'Tue',
    'Wed',
    'Thur',
    'Fri',
    'Sat'
];

return days_abbr[this.getDay()];
};

// Provide the day of year 1-365
Date.prototype.getDayOfYear = function() {
    var onejan = new Date(this.getFullYear(),0,1);
    return Math.ceil((this - onejan) / 86400000);
};

// Provide the day suffix (st,nd,rd,th)
Date.prototype.getDaySuffix = function() {
    var d = this.getDate();
    var sfx = ["th","st","nd","rd"];
    var val = d%100;

    return (sfx[(val-20)%10] || sfx[val] || sfx[0]);
};

// Provide Week of Year
Date.prototype.getWeekOfYear = function() {
    var onejan = new Date(this.getFullYear(),0,1);
    return Math.ceil((((this - onejan) / 86400000) + onejan.getDay()+1)/7);
};

// Provide if it is a Leap year or not
Date.prototype.isLeapYear = function(){
    var yr = this.getFullYear();

    if ((parseInt(yr)%4) == 0){
        if (parseInt(yr)%100 == 0){
            if (parseInt(yr)%400 != 0){
                return false;
            }
            if (parseInt(yr)%400 == 0){
                return true;
            }
        }
        if (parseInt(yr)%100 != 0){
            return true;
        }
    }
};

```

```

    }
    if ((parseInt(yr)%4) != 0){
        return false;
    }
};

// Provide Number of Days in a given month
Date.prototype.getMonthDayCount = function() {
    var month_day_counts = [
        31,
        this.isLeapYear() ? 29 : 28,
        31,
        30,
        31,
        30,
        31,
        31,
        30,
        31,
        30,
        31
    ];

    return month_day_counts[this.getMonth()];
}

// format provided date into this.format format
Date.prototype.format = function(dateFormat){
    // break apart format string into array of characters
    dateFormat = dateFormat.split("");

    var date = this.getDate(),
        month = this.getMonth(),
        hours = this.getHours(),
        minutes = this.getMinutes(),
        seconds = this.getSeconds();
    // get all date properties ( based on PHP date object functionality )
    var date_props = {
        d: date < 10 ? '0'+date : date,
        D: this.getDayAbbr(),
        j: this.getDate(),
        l: this.getDayFull(),
        S: this.getDaySuffix(),
        w: this.getDay(),
        z: this.getDayOfYear(),
        W: this.getWeekOfYear(),
        F: this.getMonthName(),
        m: month < 10 ? '0'+(month+1) : month+1,
        M: this.getMonthAbbr(),
    };

```

```

n: month+1,
t: this.getMonthDayCount(),
L: this.isLeapYear() ? '1' : '0',
Y: this.getFullYear(),
y: this.getFullYear()+'.substring(2,4),
a: hours > 12 ? 'pm' : 'am',
A: hours > 12 ? 'PM' : 'AM',
g: hours % 12 > 0 ? hours % 12 : 12,
G: hours > 0 ? hours : "12",
h: hours % 12 > 0 ? hours % 12 : 12,
H: hours,
i: minutes < 10 ? '0' + minutes : minutes,
s: seconds < 10 ? '0' + seconds : seconds
};

// Loop through format array of characters and add matching data else add the
format character (:,/, etc.)
var date_string = "";
for(var i=0;i<dateFormat.length;i++){
    var f = dateFormat[i];
    if(f.match(/[a-zA-Z]/g)){
        date_string += date_props[f] ? date_props[f] : '';
    } else {
        date_string += f;
    }
}

return date_string;
};
/*
 *
 * END - Date class extension
 *
 *****/

```

edited May 8 '13 at 18:15

answered May 6 '13 at 20:22



jmiraglia

483 3 7

OK, we have got something called **Intl** which is very useful for formatting a date in JavaScript these days:

17

Your date as below:

```
var date = '10/8/2010';
```

And you change to Date by using new Date() like below:

```
date = new Date(date);
```

And now you can format it any way you like using a list of **locales** like below:

```
date = new Intl.DateTimeFormat('en-AU').format(date); // Australian date format:  
"8/10/2010"
```

```
date = new Intl.DateTimeFormat('en-US').format(date); // USA date format: "10/8/2010"
```

```
date = new Intl.DateTimeFormat('ar-EG').format(date); // Arabic date format:  
"٨/١٠/٢٠١٠"
```

If you exactly want the format you mentioned above, you can do:

```
date = new Date(Date.UTC(2010, 7, 10, 0, 0, 0));  
var options = {year: "numeric", month: "short", day: "numeric"};  
date = new Intl.DateTimeFormat("en-AU", options).format(date).replace(/\s/g, '-');
```

And the result is going to be:

```
"10-Aug-2010"
```

For more details about **ECMAScript Internationalization API (Intl)**, visit [here](#).

edited Jan 19 at 0:34

answered Jul 19 '17 at 12:58



Alireza

59.8k

14

197

127



Not supported by IE – [Pants](#) yesterday

If you are using jQuery UI in your code, there is an inbuilt function called `formatDate()` . I am using it this way to format today's date:

15

```
var testdate = Date();
testdate = $.datepicker.formatDate( "d-M-yy", new Date(testdate));
alert(testdate);
```

You can see [many other examples of formatting date in the jQuery UI documentation](#).

edited Oct 20 '15 at 16:08



[Carrie Kendall](#)

9,619 4 53 77

answered Nov 17 '14 at 13:32



[webzy](#)

263 2 10

We have lots of solutions for this, but I think the best of them is Moment.js. So I personally suggest to use Moment.js for date and time operations.

14

```
console.log(moment().format('DD-MMM-YYYY'));
```

```
<script src="//cdnjs.cloudflare.com/ajax/libs/moment.js/2.14.1/moment.min.js"></script>
```

Run code snippet

[Expand snippet](#)

edited Mar 4 '17 at 22:33



[Peter Mortensen](#)

14.3k 19 88 116

answered Aug 29 '16 at 9:48



[Vijay Maheriya](#)

1,154 11 21

why are you including jquery? – [Ced](#) Oct 18 '16 at 17:23

1 Ohh sorry its not require. Thanks @Ced – [Vijay Maheriya](#) Oct 21 '16 at 6:47

how to provide the date i have to moment.js? I think it always takes current time. – [Dave Ranjan](#) Dec 1 '16 at 14:18

1 @DaveRanjan i think you need to convert your custom date. So use this : `console.log(moment('2016-08-10').format('DD-MMM-YYYY'))`; – [Vijay Maheriya](#) Dec 5 '16 at 5:59

Yeah, figured it out later. Thanks :) – [Dave Ranjan](#) Dec 6 '16 at 7:04

A useful and flexible way for formatting the DateTimes in JavaScript is `Intl.DateTimeFormat` :

14

```
var date = new Date();
var options = { year: 'numeric', month: 'short', day: '2-digit' };
var _resultDate = new Intl.DateTimeFormat('en-GB', options).format(date);
// The _resultDate is: "12 Oct 2017"
// Replace all spaces with - and then Log it.
console.log(_resultDate.replace(/ /g, '-'));
```

Result Is: "12-Oct-2017"

The date and time formats can be customized using the options argument.

The `Intl.DateTimeFormat` object is a constructor for objects that enable language sensitive date and time formatting.

Syntax

```
new Intl.DateTimeFormat([locales[, options]])
Intl.DateTimeFormat.call(this[, locales[, options]])
```

## Parameters

### locales

Optional. A string with a BCP 47 language tag, or an array of such strings. For the general form and interpretation of the locales argument, see the Intl page. The following Unicode extension keys are allowed:

```
nu
Numbering system. Possible values include: "arab", "arabext", "bali", "beng", "deva",
"fullwide", "gujr", "guru", "hanidec", "khmr", "knda", "lao", "latn", "limb", "mlym",
"mong", "mymr", "orya", "tamldec", "telu", "thai", "tib".
ca
Calendar. Possible values include: "buddhist", "chinese", "coptic", "ethioaa",
"ethiopic", "gregory", "hebrew", "indian", "islamic", "islamicc", "iso8601", "japanese",
"persian", "roc".
```

## Options

Optional. An object with some or all of the following properties:

### **localeMatcher**

The locale matching algorithm to use. Possible values are `"lookup"` and `"best fit"`; the default is `"best fit"`. For information about this option, see the Intl page.

### **timeZone**

The time zone to use. The only value implementations must recognize is `"UTC"`; the default is the runtime's default time zone. Implementations may also recognize the time zone names of the IANA time zone database, such as `"Asia/Shanghai"`, `"Asia/Kolkata"`, `"America/New_York"`.

### **hour12**

Whether to use 12-hour time (as opposed to 24-hour time). Possible values are `true` and `false`; the default is locale dependent.

### **formatMatcher**

The format matching algorithm to use. Possible values are `"basic"` and `"best fit"`; the default is `"best fit"`. See the following paragraphs for information about the use of this property.

The following properties describe the date-time components to use in formatted output and their desired representations. Implementations are required to support at least the following subsets:

```
weekday, year, month, day, hour, minute, second
weekday, year, month, day
year, month, day
year, month
month, day
hour, minute, second
hour, minute
```

Implementations may support other subsets, and requests will be negotiated against all available subset-representation combinations to find the best match. Two algorithms are available for this negotiation and selected by the `formatMatcher` property: A fully specified `"basic"` algorithm and an implementation dependent `"best fit"` algorithm.

### **weekday**

The representation of the weekday. Possible values are "narrow" , "short" , "long" .

**era**

The representation of the era. Possible values are "narrow" , "short" , "long" .

**year**

The representation of the year. Possible values are "numeric" , "2-digit" .

**month**

The representation of the month. Possible values are "numeric" , "2-digit" , "narrow" , "short" , "long" .

**day**

The representation of the day. Possible values are "numeric" , "2-digit" .

**hour**

The representation of the hour. Possible values are "numeric" , "2-digit" .

**minute**

The representation of the minute. Possible values are "numeric" , "2-digit" .

**second**

The representation of the second. Possible values are "numeric" , "2-digit" .

**timeZoneName**

The representation of the time zone name. Possible values are "short" , "long" . The default value for each date-time component property is undefined, but if all component properties are undefined, then the year, month and day are assumed to be "numeric" .

[Check Online](#)

[More Details](#)

edited Nov 1 '17 at 19:15

answered Oct 13 '17 at 10:58



**Iman Bahrampour**

**2,646** 2 20 45

A JavaScript solution without using any external libraries:

14

```
var now = new Date()
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
var formattedDate = now.getDate() + "-" + months[now.getMonth()] + "-" +
now.getFullYear()
alert(formattedDate)
```

edited Mar 23 at 11:44



Peter Mortensen

14.3k 19 88 116

answered Jul 21 '14 at 8:49



Prasad DLV

311 2 8

This is how I implemented for my npm plugins

13

```
var monthNames = [
  "January", "February", "March",
  "April", "May", "June", "July",
  "August", "September", "October",
  "November", "December"
];

var Days = [
  "Sunday", "Monday", "Tuesday", "Wednesday",
  "Thursday", "Friday", "Saturday"
];

var formatDate = function(dt,format){
  format = format.replace('ss', pad(dt.getSeconds(),2));
  format = format.replace('s', dt.getSeconds());
  format = format.replace('dd', pad(dt.getDate(),2));
  format = format.replace('d', dt.getDate());
  format = format.replace('mm', pad(dt.getMinutes(),2));
  format = format.replace('m', dt.getMinutes());
  format = format.replace('MMMM', monthNames[dt.getMonth()]);
  format = format.replace('MMM', monthNames[dt.getMonth()].substring(0,3));
  format = format.replace('MM', pad(dt.getMonth()+1,2));
  format = format.replace(/M(?:[ao])/, dt.getMonth()+1);
  format = format.replace('DD', Days[dt.getDay()]);
  format = format.replace(/D(?:!e)/, Days[dt.getDay()].substring(0,3));
```

```

format = format.replace('yyyy', dt.getFullYear());
format = format.replace('YYYY', dt.getFullYear());
format = format.replace('yy', (dt.getFullYear()+"").substring(2));
format = format.replace('YY', (dt.getFullYear()+"").substring(2));
format = format.replace('HH', pad(dt.getHours(),2));
format = format.replace('H', dt.getHours());
return format;
}

pad = function(n, width, z) {
  z = z || '0';
  n = n + '';
  return n.length >= width ? n : new Array(width - n.length + 1).join(z) + n;
}

```

edited Feb 23 '17 at 6:17

answered Jul 30 '16 at 16:58



Amit Kumar Gupta

4,084 9 48 72

Which package are you referring to? – [Ibrahim](#) Nov 2 '16 at 8:11

2 [date util](#) – [Amit Kumar Gupta](#) Nov 2 '16 at 9:35

This has a bug: Month names are replaced first, then the name of the month will be replaced as well. For example `March` will become `3arch` with this code. – [ntaso](#) Feb 22 '17 at 9:41

1 Change line for `'M'` to `format = format.replace("M(?:!M)", (dt.getMonth()+1).toString());` and put it above line with `'MMMM'` – [ntaso](#) Feb 22 '17 at 9:46

13

```
new Date().toLocaleDateString()
```

```
// "3/21/2018"
```

Run code snippet

[Expand snippet](#)

More documentation at [developer.mozilla.org](https://developer.mozilla.org)

edited Mar 21 '18 at 17:23

answered Jan 15 '18 at 2:33



Kirk Strobeck

10.8k 13 58 96

1 Should be noted that you should never ever, ever, use document.write(). Huge security and performance issues – Eugene Fidelin Feb 14 '18 at 15:39

This may help with the problem:

13

```
var d = new Date();

var options = {
  day: 'numeric',
  month: 'long',
  year: 'numeric'
};

console.log(d.toLocaleDateString('en-ZA', options));
```

[Run code snippet](#)[Expand snippet](#)

```
1 var d = new Date();
2 var options = { day: 'numeric', month: 'long', year: 'numeric'};
3 d.toLocaleDateString('en-ZA', options);
```

edited Mar 23 at 11:48



Peter Mortensen

14.3k 19 88 116

answered Jan 8 '18 at 10:27



Mr Nsubuga

174 1 7

1 or d.toLocaleDateString('en-US', options); if you are in the USA. – BishopZ Jan 27 '18 at 6:31

This was my solution. Thank you. – Steven Rogers Apr 5 at 19:05

```
var today = new Date();
```

9

```
var formattedToday = today.toLocaleDateString() + ' ' + today.toLocaleTimeString();
```

edited Jan 18 '17 at 20:22



Jeffrey Knight

4,612 5 33 45

answered Feb 9 '16 at 22:00



Gennadiy Ryabkin

5,643 3 24 34

Sugar.js has excellent extensions to the Date object, including a [Date.format](#) method.

8

Examples from the documentation:

```
Date.create().format('{Weekday} {Month} {dd}, {yyyy}');
```

```
Date.create().format('{12hr}:{mm}{tt}')
```

edited May 18 '15 at 15:56



Peter Mortensen

14.3k 19 88 116

answered Oct 22 '12 at 22:10



hasen

86.5k 60 170 213

For any one looking for a really simple ES6 solution to copy, paste and adopt:

7

```
const dateToString = d => `${d.getFullYear()}-${('00' + (d.getMonth() + 1)).slice(-2)}-${('00' + d.getDate()).slice(-2)}`
```

*// how to use:*

```
const myDate = new Date(Date.parse('04 Dec 1995 00:12:00 GMT'))  
console.log(dateToString(myDate)) // 1995-12-04
```

[Run code snippet](#)[Expand snippet](#)

edited Nov 26 '18 at 9:08

answered May 7 '18 at 15:00



Hinrich

8,335 3 23 42



▲ Add the [jQuery UI](#) plugin to your page:

4

```
function DateFormat(dateFormat, datetime) {
    return $.datepicker.formatDate(dateFormat, datetime);
};
```

edited May 18 '15 at 15:55



Peter Mortensen

14.3k 19 88 116

answered Jun 29 '12 at 12:02



Thulasiram

7,227 6 37 41

13 Don't add jQuery and jQuery UI just to format a date! – [Bennett McElwee](#) Sep 9 '15 at 3:05

2 if you already using JQuery UI in your project it could be a good approach, but it seems you can only show the date portion , no hours, minutes or seconds – [A.Alqadomi](#) Sep 21 '15 at 9:36

2 why not plain jquery? \$.format.date(new Date(), 'yyyy-MM-dd HH:mm:ss') – [Alexander](#) Feb 3 '16 at 13:25

▲ Try this:

4

```
function init(){
    var d = new Date();
    var day = d.getDate();
    var x = d.toString().substr(4, 3);
    var year = d.getFullYear();
    document.querySelector("#mydate").innerHTML = day + '-' + x + '-' + year;
}
window.onload = init;
```

<div id="mydate"></div>

Run code snippet

[Expand snippet](#)

edited Mar 4 '17 at 22:28



Peter Mortensen

14.3k 19 88 116

answered Dec 31 '15 at 10:01



Arjun Nayak

685 7 17



```
DateFormatter.formatDate(new Date(2010,7,10), 'DD-MMM-YYYY')
```

4

```
=> 10-Aug-2010
```



```
DateFormatter.formatDate(new Date(), 'YYYY-MM-DD HH:mm:ss')
```

```
=> 2017-11-22 19:52:37
```

```
DateFormatter.formatDate(new Date(2005, 1, 2, 3, 4, 5), 'D DD DDD DDDD, M MM MMM MMMM, YY YYYY, h hh H HH, m mm, s ss, a A')
```

```
=> 2 02 Wed Wednesday, 2 02 Feb February, 05 2005, 3 03 3 03, 4 04, 5 05, am AM
```

```
var DateFormatter = {
  monthNames: [
    "January", "February", "March", "April", "May", "June",
    "July", "August", "September", "October", "November", "December"
  ],
  dayNames: ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
    "Saturday"],
  formatDate: function (date, format) {
    var self = this;
    format = self.getProperDigits(format, /d+/gi, date.getDate());
    format = self.getProperDigits(format, /M+/g, date.getMonth() + 1);
    format = format.replace(/y+/gi, function (y) {
      var len = y.length;
      var year = date.getFullYear();
      if (len == 2)
        return (year + "").slice(-2);
      else if (len == 4)
        return year;
      return y;
    })
    format = self.getProperDigits(format, /H+/g, date.getHours());
    format = self.getProperDigits(format, /h+/g, self.getHours12(date.getHours()));
    format = self.getProperDigits(format, /m+/g, date.getMinutes());
    format = self.getProperDigits(format, /s+/gi, date.getSeconds());
    format = format.replace(/a/ig, function (a) {
      var amPm = self.getAmPm(date.getHours())
      if (a === 'A')
        return amPm.toUpperCase();
      return amPm;
    })
    format = self.getFullOr3Letters(format, /d+/gi, self.dayNames, date.getDay())
  }
}
```

```

    format = self.getFullOr3Letters(format, /M+/g, self.monthNames, date.getMonth());
    return format;
  },
  getProperDigits: function (format, regex, value) {
    return format.replace(regex, function (m) {
      var length = m.length;
      if (length == 1)
        return value;
      else if (length == 2)
        return ('0' + value).slice(-2);
      return m;
    })
  },
  getHours12: function (hours) {
    // https://stackoverflow.com/questions/10556879/changing-the-1-24-hour-to-1-12-hour-
    // for-the-gethours-method
    return (hours + 24) % 12 || 12;
  },
  getAmPm: function (hours) {
    // https://stackoverflow.com/questions/8888491/how-do-you-display-javascript-
    // datetime-in-12-hour-am-pm-format
    return hours >= 12 ? 'pm' : 'am';
  },
  getFullOr3Letters: function (format, regex, nameArray, value) {
    return format.replace(regex, function (s) {
      var len = s.length;
      if (len == 3)
        return nameArray[value].substr(0, 3);
      else if (len == 4)
        return nameArray[value];
      return s;
    })
  }
}

console.log(DateFormatter.formatDate(new Date(), 'YYYY-MM-DD HH:mm:ss'));
console.log(DateFormatter.formatDate(new Date(), 'D DD DDD DDDD, M MM MMM MMMM, YY YYYY,
h hh H HH, m mm, s ss, a A'));
console.log(DateFormatter.formatDate(new Date(2005, 1, 2, 3, 4, 5), 'D DD DDD DDDD, M MM
MMM MMMM, YY YYYY, h hh H HH, m mm, s ss, a A'));

```

Run code snippet

[Expand snippet](#)

The format description was taken from [Ionic Framework](#) (it does not support `z`, UTC Timezone Offset)

Not thoroughly tested

edited Mar 23 at 11:46



Peter Mortensen

14.3k 19 88 116

answered Nov 22 '17 at 14:24



amit77309

4,290 3 12 21