

Dates in JavaScript

An Introduction

Web Development 1

John Rellis

Is this a date?

2022-03-01

1646092800000

03-01-2022

2022-03-01T00:00:00+00:00

Tuesday, March 1, 2022



Is this a date?

Yes, these all represent the same date

2022-03-01

1646092800000

2022-03-01T00:00:00+00:00

03-01-2022

Tuesday, March 1, 2022



Working with Dates in JavaScript

- JavaScript has a built in Date class
- It's awful
-
-
- Well, not too bad, but we won't be using it too much

```
> new Date()
< Wed Apr 24 2024 13:24:42 GMT+0100 (Irish Standard Time)
```

```
> const now = new Date()
```

```
< undefined
```

```
> now.valueOf()
```

```
< 1713961489609
```

```
> |
```

The secret to working with dates

- Dates are merely a value in milliseconds since the epoch
- Epoch is midnight at the beginning of January 1, 1970, UTC
- Essentially, dates are a wrapper on a millisecond value
- These can then be formatted to calendar notation with or without timezones
- When working with dates as strings, you should always include a timezone
- But we are going to skim the surface, since, well, it gets complicated

```
> new Date(0)
<- Thu Jan 01 1970 01:00:00 GMT+0100 (Greenwich Mean Time)

> new Date(1000)
<- Thu Jan 01 1970 01:00:01 GMT+0100 (Greenwich Mean Time)

> new Date()
<- Wed Apr 24 2024 13:31:39 GMT+0100 (Irish Standard Time)

> new Date().valueOf()
<- 1713961906274
```

Date time string format

- There are many ways to format a date as a string.
- The JavaScript specification only specifies one format to be universally supported: the date time string format, a simplification of the ISO 8601 calendar date extended format.
- The format is as follows:
 - YYYY-MM-DDTHH:mm:ss.sssZ

Date time string format

- YYYY-MM-DDTHH:mm:ss.sssZ
- YYYY is the year, with four digits (0000 to 9999), or as an expanded year of + or - followed by six digits. The sign is required for expanded years. -000000 is explicitly disallowed as a valid year.
- MM is the month, with two digits (01 to 12). Defaults to 01.
- DD is the day of the month, with two digits (01 to 31). Defaults to 01.
- T is a literal character, which indicates the beginning of the time part of the string. The T is required when specifying the time part.
- HH is the hour, with two digits (00 to 23). As a special case, 24:00:00 is allowed and is interpreted as midnight at the beginning of the next day. Defaults to 00.
- mm is the minute, with two digits (00 to 59). Defaults to 00.
- ss is the second, with two digits (00 to 59). Defaults to 00.
- sss is the millisecond, with three digits (000 to 999). Defaults to 000.
- Z is the timezone offset, which can either be the literal character Z (indicating UTC), or + or - followed by HH:mm, the offset in hours and minutes from UTC.
- <https://tc39.es/ecma262/multipage/numbers-and-dates.html#sec-date-time-string-format>

```
Date.parse("2024-01-01T05:06:12.123")
1704085572123
new Date("2024-01-01T05:06:12.123")
Mon Jan 01 2024 05:06:12 GMT+0000 (Greenwich Mean Time)
```

The API

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

The screenshot shows the MDN Web Docs page for the `Date` object. The page has a dark theme with a navigation bar at the top featuring links for References, Guides, Plus, Curriculum (NEW), Blog, Play, AI Help (BETA), Theme, Log in, and Sign up for free. The main content area shows the `Date` object under the Standard built-in objects category. The `Date` constructor and various methods like `getDate()`, `getDay()`, etc., are listed. A note box states: "Note: TC39 is working on Temporal, a new Date/Time API. Read more about it on the Igalia blog. It is not yet ready for production use!" Below this is a

Description

 section with text about the epoch, timestamps, and invalid date. Another note box says: "Note: While the time value at the heart of a Date object is UTC, the basic methods to fetch the date and time or its components all work in the local (i.e. host system) time zone and offset." At the bottom, there's a note about the maximum timestamp representable by a `Date` object.

mdn web docs

References Guides Plus Curriculum NEW Blog Play AI Help BETA

Theme Log in Sign up for free

English (US)

References > JavaScript > Reference > Standard built-in objects > Date

Filter

Date

Standard built-in objects

`Date`

Constructor

`Date()` constructor

Methods

`Date.prototype[@toPrimitive]()`

`Date.prototype.getDate()`

`Date.prototype.getDay()`

`Date.prototype.getFullYear()`

`Date.prototype.getHours()`

`Date.prototype.getMilliseconds()`

`Date.prototype.getMinutes()`

`Date.prototype.getMonth()`

`Date.prototype.getSeconds()`

`Date.prototype.getTime()`

`Date.prototype.getTimezoneOffset()`

`Date.prototype.getUTCDate()`

`Date.prototype.getUTCDay()`

`Date.prototype.getUTCFullYear()`

`Date.prototype.getUTCHours()`

`Date` objects represent a single moment in time in a platform-independent format. `Date` objects encapsulate an integral number that represents milliseconds since the midnight at the beginning of January 1, 1970, UTC (the *epoch*).

Note: TC39 is working on [Temporal](#), a new Date/Time API. Read more about it on the [Igalia blog](#). It is not yet ready for production use!

Description

The epoch, timestamps, and invalid date

A JavaScript date is fundamentally specified as the time in milliseconds that has elapsed since the [epoch](#), which is defined as the midnight at the beginning of January 1, 1970, UTC (equivalent to the [UNIX epoch](#)). This timestamp is *timezone-agnostic* and uniquely defines an instant in history.

Note: While the time value at the heart of a Date object is UTC, the basic methods to fetch the date and time or its components all work in the local (i.e. host system) time zone and offset.

The maximum timestamp representable by a `Date` object is slightly smaller than the maximum safe integer (`Number.MAX_SAFE_INTEGER`, which is 9,007,199,254,740,991). A `Date` object can represent a maximum of ±8,640,000,000,000,000 milliseconds, or ±100,000,000 (one hundred million) days,

In this article

- Description
- Constructor
- Static methods
- Instance properties
- Instance methods
- Examples
- Specifications
- Browser compatibility
- See also

Why don't you like the JS Date?

Wed Apr 24 2024 13:47:19 GMT+0100 (Irish Standard Time)

```
date.getYear()  
124 ←  
date.getFullYear()  
2024  
date.getDay()  
3 ←  
date.getDate()  
24  
date.getSeconds()  
19
```

What?

And numerous other annoyances

What?

So.....?

```
dayjs('2018-08-08') // parse  
  
dayjs().format('{YYYY} MM-DDTHH:mm:ss SSS [Z] A') // display  
  
dayjs().set('month', 3).month() // get & set  
  
dayjs().add(1, 'year') // manipulate  
  
dayjs().isBefore(dayjs()) // query
```

dayjs TS
1.11.10 • Public • Published 7 months ago

[Readme](#) [Code](#) Beta [0 Dependencies](#) [16,541 Dependents](#) [123 Versions](#)

English | 简体中文 | 日本語 | Português Brasileiro | 한국어 | Español (España) | Русский | Türkçe
| தமிழ் | עברית

DAY.JS

Fast 2kB alternative to Moment.js with the same modern API

The screenshot shows the GitHub repository page for Day.js. At the top, there's a large red "DAY.JS" logo. Below it, the README file is displayed with code examples. To the right, there's a "Contributors" section showing 328 contributors with small profile pictures, and a "Deployments" section showing a successful deployment to GitHub Pages. The "Languages" section indicates the code is written in 100% JavaScript. In the center, there's a "Gzip Size" badge for npm v1.11.10, build status, coverage (100%), and license (MIT). Below the badge, there's a compatibility table for various mobile devices and a brief description of the library's purpose. At the bottom, there are sections for "Sponsor", "Watch", "Fork", "Star", and "About". The "About" section contains a summary of the library.

README MIT license

English | 简体中文 | 日本語 | Português Brasileiro | 한국어 | Español (España) | Русский | Türkçe | தமிழ் | עברית

DAY.JS

Fast 2kB alternative to Moment.js with the same modern API

Gzip Size npm v1.11.10 build passing coverage 100% license MIT

	iPhone	Edge	Safari
7	✓ 9.3 X 10.11 ✓	13	10 ✓ 10.10 ✓
10	11 X 10.12 ✓	17	10 ✓ 11 X u ✓

dates, manipulates, and displays dates and times for API. If you use Moment.js, you already know how to

Sponsor Watch 282 Fork 2.2k Star 45.7k

0309582 · 19 hours ago 1,513 Commits

last year

Day.js 2kB immutable date-time library alternative to Moment.js with the same modern API

<https://day.js.org/docs/en/installation/installation>

The screenshot shows the GitHub repository page for Day.js. At the top, there's a navigation bar with the repository name "Day.js", a star icon, and the number "45,744". To the right are links for "Search", "Docs", "GitHub", and language selection ("English"). Below the navigation, the main content area has a sidebar on the left with a tree view of the documentation structure. The main content area is titled "Browser" and contains code snippets for installation and CDN inclusion, along with explanatory text and links to related sections.

Installation

Installation

Node.js

Browser

TypeScript

Download

Parse

Get + Set

Manipulate

Display

Query

i18n

Plugins

Customize

Durations

Time Zone

Browser

[EDIT](#)

```
<script src="path/to/dayjs/dayjs.min.js"></script>
<script>
  dayjs().format()
</script>
```

CDN resource

Day.js can be included by way of a CDN provider like cdnjs.com, [unpkg](https://unpkg.com) and [jsDelivr](https://jsDelivr.com) ...

```
<!-- CDN example (jsDelivr) -->
<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>
<script>dayjs().format()</script>
```

Check here for more information about loading [locale](#) and [plugin](#).

[← NODE.JS](#)

[TYPESCRIPT →](#)

Installation >

Now

EDIT

Parse <

Parse

Now

String

String + Format

Unix Timestamp (milliseconds)

Unix Timestamp (seconds)

Date

Object

Array

UTC

Dayjs Clone

Validation

Get + Set >

Manipulate >

Display >

Query >

i18n >

Plugins >

```
var now = dayjs()
```

This is essentially the same as calling `dayjs(new Date())`.

Dayjs treats `dayjs(undefined)` as `dayjs()` due to that function parameters default to undefined when not passed in.

Dayjs treats `dayjs(null)` as an invalid input.

← PARSE

STRING →

Installation



String

[EDIT](#)

Parse



Parse the given string in [ISO 8601](#) format (a space instead of the 'T' is allowed) and return a Day.js object instance.

Parse

Now

String

String + Format

Unix Timestamp (milliseconds)

Unix Timestamp (seconds)

Date

Object

Array

UTC

Dayjs Clone

Validation

Get + Set



Manipulate



Display



Query



i18n



Plugins



Customize



```
dayjs('2018-04-04T16:00:00.000Z')
dayjs('2018-04-13 19:18:17.040+02:00')
dayjs('2018-04-13 19:18')
```

For consistent results parsing anything **other** than ISO 8601 strings, you should use [String + Format](#).

[← NOW](#)[STRING + FORMAT →](#)

Installation



Parse



Get + Set



Manipulate



Display



Display

Format

Time from now

Time from X

Time to now

Time to X

Calendar Time

Difference

Unix Timestamp (milliseconds)

Unix Timestamp

Days in Month

As Javascript Date

As Array

As JSON

As ISO 8601 String

As Object

As String

Query



i18n



Plugins



Customize



Durations



Format

EDIT

Get the formatted date according to the string of tokens passed in.

To escape characters, wrap them in square brackets (e.g. `[MM]`).

```
dayjs().format()  
// current date in ISO8601, without fraction seconds e.g. '2020-04-02T08:02:17-05:00'  
  
dayjs('2019-01-25').format('[YYYYescape] YYYY-MM-DDTHH:mm:ssZ[Z]')  
// 'YYYYescape 2019-01-25T00:00:00-02:00Z'  
  
dayjs('2019-01-25').format('DD/MM/YYYY') // '25/01/2019'
```

List of all available formats

Format	Output	Description
YY	18	Two-digit year
YYYY	2018	Four-digit year
M	1-12	The month, beginning at 1
MM	01-12	The month, 2-digits
MMM	Jan-Dec	The abbreviated month name
MMMM	January-December	The full month name
D	1-31	The day of the month
DD	01-31	The day of the month, 2-digits
d	0-6	The day of the week, with Sunday as 0
dd	Su-Sa	The min name of the day of the week
ddd	Sun-Sat	The short name of the day of the week
dddd	Sunday-Saturday	The name of the day of the week
H	0-23	The hour



Installation



Add

[EDIT](#)

Parse



Returns a cloned Day.js object with a specified amount of time added.

Get + Set



```
const a = dayjs()
```

```
const b = a.add(7, 'day')
```

Manipulate



Manipulate

[Add](#)

Subtract

Start of Time

End of Time

Local

UTC

UTC offset

Display



Query



i18n



Plugins



Customize



Durations



Time Zone



Units are case insensitive, and support plural and short forms. Note, short forms are case sensitive.

List of all available units

Unit	Shorthand	Description
day	d	Day
week	w	Week
month	M	Month
quarter	Q	Quarter (dependent QuarterOfYear plugin)
year	y	Year
hour	h	Hour
minute	m	Minute
second	s	Second
millisecond	ms	Millisecond

Alternatively, you can use [durations](#) to add to Day.js object.

```
result = dayjs().add(dayjs.duration({'days' : 1}))
```

When decimal values are passed for **days** and **weeks**, they are rounded to the nearest integer before adding.

```
const now = dayjs(); // create a new date object that represents this time
const currentHour = now.hour(); // 0 - 23 as a number

console.log(currentHour) // 14 as a number
console.log(now.format("HH")) // 14 string - https://day.js.org/docs/en/display/format
console.log(now.format("hh")) // 02 as a string - 12 hour format
console.log(now.format("dddd")); // Wednesday
console.log(now.format("ddd")); // Wed

console.log( now.add(1, 'day').format("dddd")); // Thursday - getting tomorrow
console.log(now.add(2, 'day').format("dddd")); // Friday - getting day after tomorrow

console.log(now.second()) // https://day.js.org/docs/en/get-set/second
console.log(now.hour()) // https://day.js.org/docs/en/get-set/hour
```

The day today is Wednesday

The day tomorrow is Thursday

The day after tomorrow is Friday

The weather code for today is 3

The weather code for tomorrow is 1

The current hour is 14:00

The current temp is 8.3

The mex temp is for today is 8.7

The temp at the same time tomorrow is 9.6

```
<body>
  <p>The day today is <span id="day"></span></p>
  <p>The day tomorrow is <span id="tomorrow"></span></p>
  <p>The day after tomorrow is <span id="dayAfterTomorrow"></span></p>
  <p>The weather code for today is <span id="weatherCodeToday"></span></p>
  <p>The weather code for tomorrow is <span id="weatherCodeTomorrow"></span></p>
  <p>The current hour is <span id="time"></span></p>
  <p>The current temp is <span id="temp"></span></p>
  <p>The mex temp is for today is <span id="maxTemp"></span></p>
  <p>The temp at the same time tomorrow is <span id="tempTomorrow"></span></p>
</body>
```

```
<script src="lab_weather_data.js"></script>
<script src="https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js"></script>
<script>
  document.addEventListener("DOMContentLoaded", () => [
    const urlParams = new URLSearchParams(window.location.search);
    const currentCity = urlParams.get('city');
    const currentCityData = weatherData[currentCity + "_daily"].daily;
    const hourlyData = weatherData[currentCity + "_hourly"].hourly;

    const weatherCodeToday = document.getElementById("weatherCodeToday");
    const weatherCodeTomorrow = document.getElementById("weatherCodeTomorrow");

    weatherCodeToday.innerHTML = currentCityData.weather_code[0];
    weatherCodeTomorrow.innerHTML = currentCityData.weather_code[1];

    const now = dayjs(); // create a new date object that represents this time
    const currentHour = now.hour(); // 0 - 23 as a number
    const indexOfCurrentHour = hourlyData.time.indexOf(`TodayT${currentHour}:00`);
    const indexOfCurrentHourTomorrow = hourlyData.time.indexOf(`Today+1T${currentHour}:00`);

    console.log(`The index of the current hour is : ${indexOfCurrentHour}`);
    console.log(`The index of the current hour tomorrow is : ${indexOfCurrentHourTomorrow}`);

    const time = document.getElementById("time");
    const temp = document.getElementById("temp");

    const tempTomorrow = document.getElementById("tempTomorrow");
    const dayAfterTomorrow = document.getElementById("dayAfterTomorrow");

    const day = document.getElementById("day");
    const tomorrow = document.getElementById("tomorrow");

    time.innerHTML = `${currentHour}:00`;
    day.innerHTML = now.format("ddd");
    tomorrow.innerHTML = now.add(1, 'day').format("ddd");
    dayAfterTomorrow.innerHTML = now.add(2, 'day').format("ddd");

    temp.innerHTML = hourlyData.temperature_2m[indexOfCurrentHour];
    tempTomorrow.innerHTML = hourlyData.temperature_2m[indexOfCurrentHourTomorrow];

    const todayMaxTemp = currentCityData.temperature_2m_max[0];
    const maxTemp = document.getElementById("maxTemp");
    maxTemp.innerHTML = todayMaxTemp;
  ]);
</script>
```