Dates in JavaScript

An Introduction

Web Development 1

John Rellis

Is this a date?

2022-03-01

1646092800000

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

03-01-2022

Tuesday, March 1, 2022

Is this a date?

Yes, these all represent the same date

2022-03-01

1646092800000

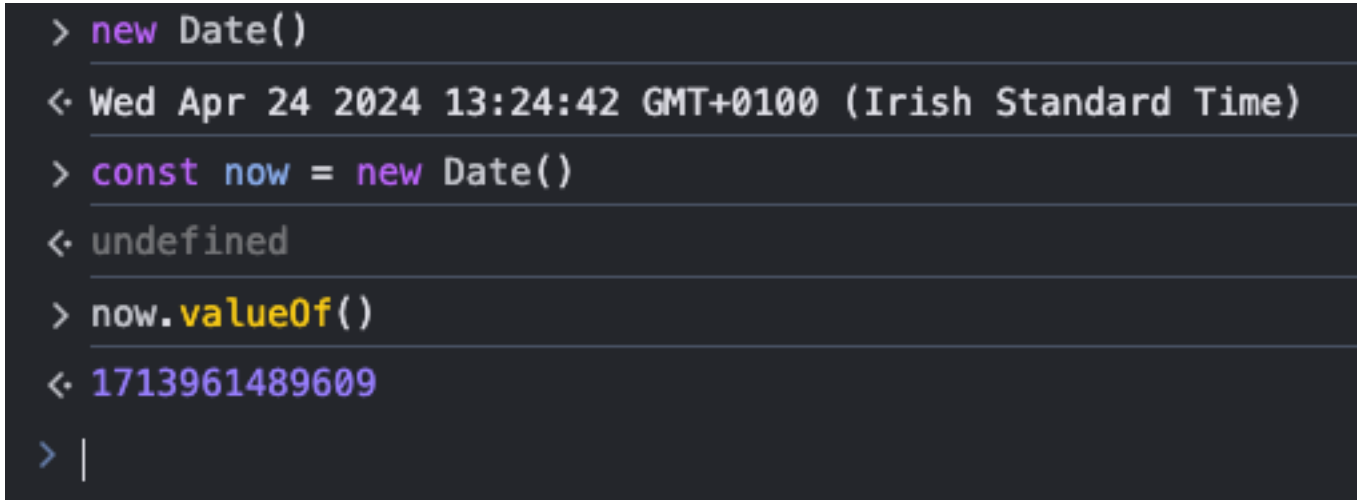
2022-03-01T00:00:00+00:00

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



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

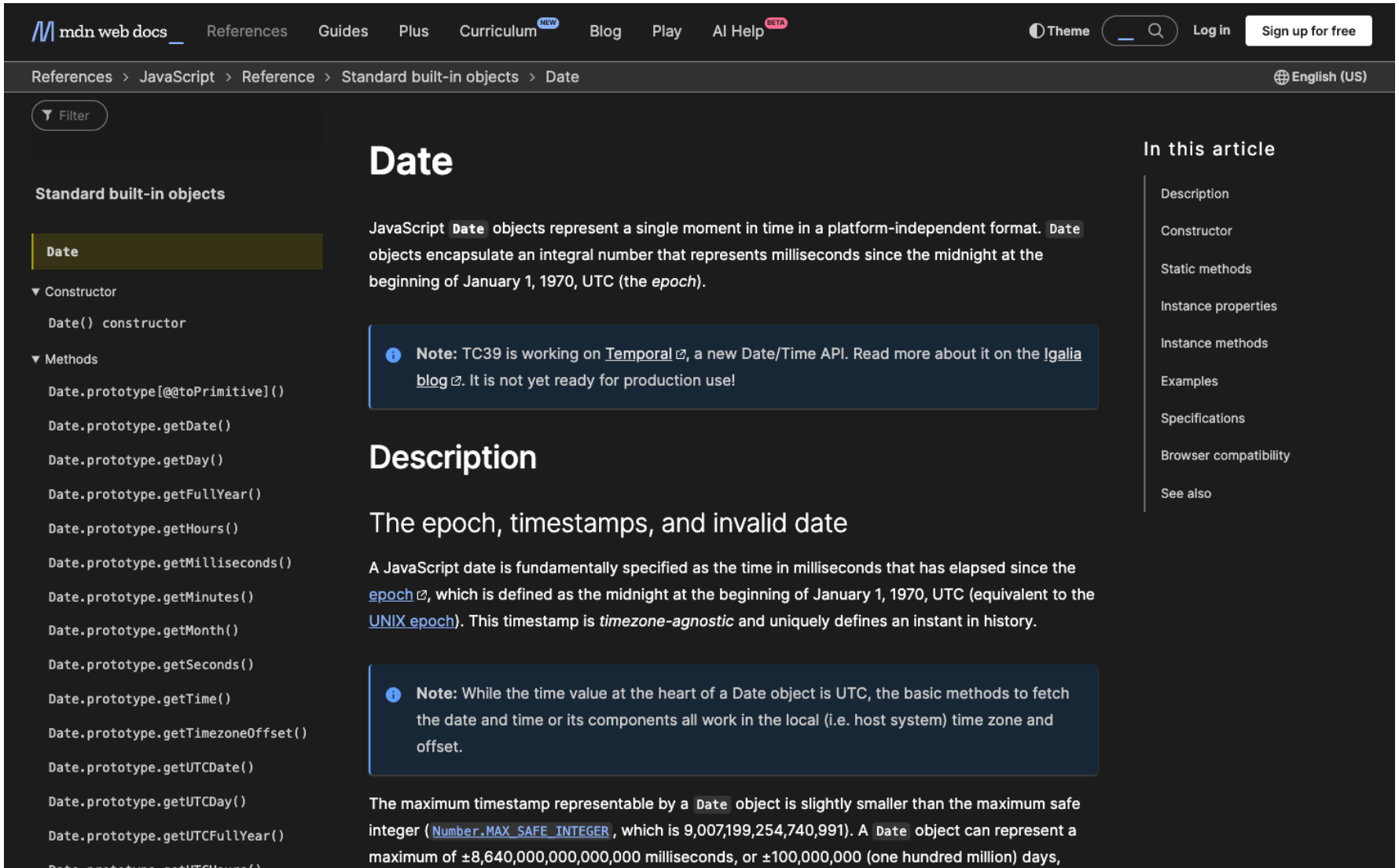
Date time string format

- 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



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?

What?

And numerous other annoyances

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
```

README | MIT license

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

9.3 10.11 | 13 10 | 8 10.10

11 10.12 | 17 10 | 11 u

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

+ 2,618,870

Contributors 328

+ 314 contributors

Deployments 10

github-pages 4 years ago

+ 9 deployments

Languages

JavaScript 100.0%

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 | සිංහල | עברית

DAY.JS

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

Install

```
> npm i dayjs
```

Repository

github.com/iamkun/dayjs

Homepage

day.js.org

Weekly Downloads

18,126,446

Sponsor | Watch 282 | Fork 2.2k | Star 45.7k

Add file | Code

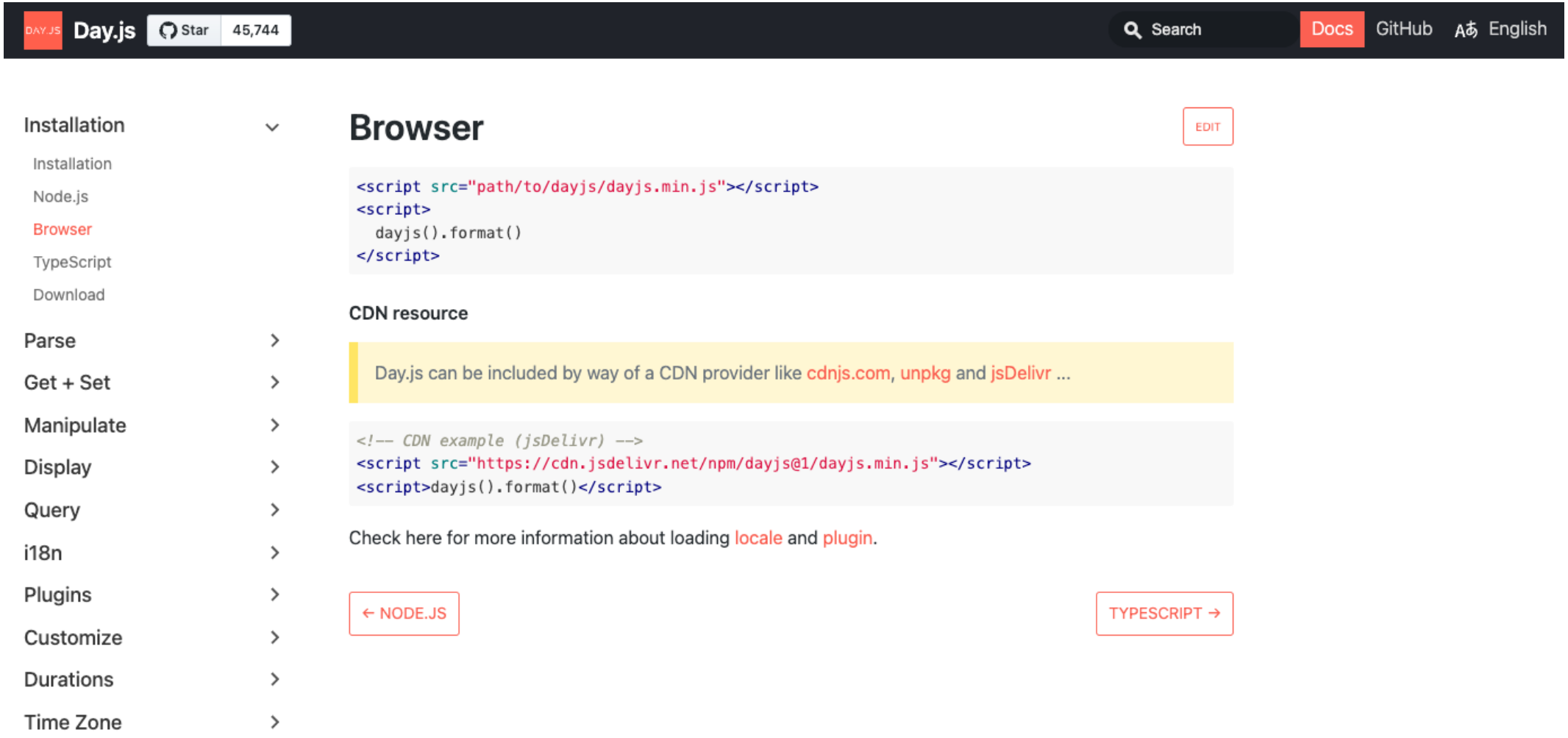
0309582 · 19 hours ago | 1,513 Commits

last year

About

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

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



Now

EDIT

Calling `dayjs()` without parameters returns a fresh Day.js object with the current date and time.

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

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

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

Day.js treats `dayjs(null)` as an invalid input.

← PARSE

STRING →

String

EDIT

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

```
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 →

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

Add

EDIT

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

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

// a -> the original value and will not change
// b -> the manipulation result
```

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

```html
<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>
```

```html
<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      You, 1 minute ago • Uncommitted changes

        // we need the index of the current hour in the time array
        // we then use that index to query the termperature_2m data
        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("dddd");
        tomorrow.innerHTML = now.add(1, 'day').format("dddd");
        dayAfterTomorrow.innerHTML = now.add(2, 'day').format("dddd");

        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>
```