It's about time!

The new Temporal API

andrena
OBJECTS

Marco Sieben

- → Working in software development at andrena for nearly 10 years
- → Preference for Frontend and TypeScript
- → Regularly found at Munich Frontend Meetups
- → Has frequently dealt with time and date specifications in various client projects

andrena
OBJECTS

We already have Date

🤔

andrena
OBJECTS

Quiztime!

```
new Date(0)
```

→ Thu Jan 01 1970 01:00:00 GMT+0100

```
new Date("0")
```

→ Sat Jan 01 2000 00:00:00 GMT+0100

```
new Date("1")
```

→ Mon Jan 01 2001 00:00:00 GMT+0100

```
new Date("2")
```

→ Thu Feb 01 2001 00:00:00 GMT+0100

https://jsdate.wtf for more

andrena OBJECTS

But wait! There's more!

```
new Date(2026, 2, 19)
```

→ Thu Mar 19 2026 00:00:00 GMT+0100

```
const date = new Date(2026, 2, 19)
date.setYear(2025)
date
```

→ Wed Mar 19 2025 00:00:00 GMT+0100

```
const date = new Date('2026-03-29T01:00:00')
date.toISOString()
```

→ 2026-03-29T00:00:00.000Z

```
date.setHours(date.getHours() + 2)
date.toISOString()
```

→ 2026-03-29T01:00:00.000Z

Well, at least Date can handle time zones

- → Not really, it only shows the local offset
- → Offset !== Time zone
- → Cannot have ***no*** offset
- → Always date and time

Challenges with Time/Date

- → Exact point in time
- → Time with time zone
- → Only date/time, without zone

Fine, then I'll just use moment/luxon/Day.js

- **Moment.js**
  - Maintenance mode
  - mutable
  - large and not tree-shakable
- **luxon, date-fns, Day.js**
  - External dependency
  - No PlainDate/PlainTime

Temporal to the rescue!

- → Immutable
- → Type safe
- → Fail fast
- → ISO-8601 compliant

andrena
OBJECTS

Good things take time

- **2017:** Proposal introduced
- **September 2018:** Stage 1
- **January 2019:** Stage 2
- **March 2021:** Stage 3
- **May 2025:** Support in Firefox 139
- **January 2026:** Support in Chrome 144
- **March 2026:** Stage 4
- **May 2026:** Support in Node 26

Why so long?

- → Adapted to global standards (ISO 8601) and IETF extensions (RFC 9557)
- → Nanoseconds instead of milliseconds
- → Plain vs Civil vs Local

Current Support

Firefox	✔️	since May 2025
Chrome	✔️	since January 2026
Safari	🧪	
Opera	❌	
Deno	✔️	since March 2026
Node	✔️	since May 2026
Bun	❌	
TypeScript	✔️	since March 2026

Polyfills are available for unsupported engines

Concepts

- **Exact point in time**
  - `Instant`
- **Time with time zone**
  - `ZonedDateTime`
- **Plain Time**
  - `PlainDate`
  - `PlainTime`
  - `PlainDateTime`
  - `PlainMonthDay`
  - `PlainYearMonth`
- **Time differences**
  - `Duration`

Usage

andrena
OBJECTS

Now

```
Temporal.Now.instant();
Temporal.Now.zonedDateTimeISO();
Temporal.Now.plainDateTimeISO();
Temporal.Now.plainDateISO();
Temporal.Now.plainTimeISO();
Temporal.Now.timeZoneId();
```

Specific Data

```
new Temporal.PlainDate(2020, 7, 21);
new Temporal.PlainTime(12, 45, 17, 88, 91, 3);
new Temporal.PlainDateTime(2020, 7, 21, 12, 45, 17, 88, 91, 3);
new Temporal.Instant(1595365932000000000n);
new Temporal.ZonedDateTime(1595365932000000000n, 'Europe/Berlin');
new Temporal.Duration(5, 2, 3, 1, 6, 7, 4, 8, 20, 54);
```

Forget the constructors right away!

Factory Methods

```
Temporal.PlainDate.from({ year: 2020, month: 7, day: 21 });
Temporal.PlainTime.from({ hour: 12, minute: 45 });
Temporal.PlainDateTime.from({ year: 2020, month: 7, day: 21, hour: 12, minute: 45 })
Temporal.Instant.fromEpochMilliseconds(1595365932000);
Temporal.Instant.from('2026-05-07T12:34:00Z');
Temporal.ZonedDateTime.from({ year: 2020, month: 7, day: 21, timeZone: 'Europe/Berli
Temporal.Duration.from({ years: 5, days: 2, minutes: 3 });
```

Parsing

```
Temporal.PlainDate.from('2020-07-21');
Temporal.PlainTime.from('12:45');
Temporal.PlainDateTime.from('2020-07-21T12:45');
Temporal.Instant.from('2026-05-07T12:34:00Z');
Temporal.ZonedDateTime.from('2020-07-21[Europe/Berlin]');
Temporal.Duration.from('P5Y2DT3M');
```

Quiztime 2 – Electric Boogaloo

```
Temporal.PlainDate.from('2020-07-21T13:32:04');
Temporal.PlainDate.from('2020-07-21T13:32:04Z');
Temporal.PlainDate.from('2020-07-21T13:32:04+02:00');
```

→ ✔️ / ❌ / ✔️

```
Temporal.PlainDateTime.from({ year: 2020, month: 7, day: 1, minute: 13 });
Temporal.PlainDateTime.from({ year: 2020, month: 7, minute: 13 });
```

→ ✔️ / ❌

```
Temporal.ZonedDateTime.from('2020-07-21T01:03:23+01:00');
Temporal.ZonedDateTime.from('2020-07-21T01:03:23+01:00[Europe/Berlin]');
Temporal.ZonedDateTime.from('1979-07-21T01:03:23+01:00[Europe/Berlin]');
Temporal.ZonedDateTime.from('2020-07-21T01:03:23[Europe/Berlin]');
Temporal.ZonedDateTime.from('2020-07-21T01:03:23Z[Europe/Berlin]');
Temporal.Instant.from('2020-07-21T01:03:23[Europe/Berlin]');
Temporal.Instant.from('2020-07-21T01:03:23+01:00[Europe/Berlin]');
```

→ ❌ / ❌ / ✔️ / ✔️ / ✔️ / ❌ / ✔️

To ISO Strings

```
somePlainDate.toString(); // 2020-07-21
somePlainTime.toString(); // 12:45:00
somePlainDateTime.toString(); // 2020-07-21T12:45:00
someInstant.toString(); // 2026-05-07T12:34:00Z
someZonedDateTime.toString(); // 2020-07-21T00:00:00+02:00[Europe/Berlin]
someZonedDateTime.toString({ timeZoneName: 'never' } ); // 2020-07-21T00:00:00+02:00
someDuration.toString(); // P5Y2DT3M
```

To Locale Strings

```
somePlainDate.toLocaleString('en'); // 7/21/2020
somePlainDate.toLocaleString('en', { dateStyle: 'long' }); // July 21, 2020
somePlainDate.toLocaleString('en', { year: '2-digit', month: 'long' }); // July 20
someZonedDateTime.toLocaleString('en'); // 7/21/2020, 11:12:12 PM GMT+2
```

Manipulation

```
const somePlainDate = Temporal.PlainDate.from({ year: 2020, month: 7, day: 21 });
somePlainDate.add({ days: 12 }); // ⇒ 2020-08-02
somePlainDate.subtract({ days: 25 }); // ⇒ 2020-06-26
somePlainDate.with({ day: 1 }); // ⇒ 2020-07-01
somePlainDate.equals(otherPlainDate);
Temporal.PlainDate.compare(somePlainDate, otherPlainDate);
```

Manipulation

```
const zdt = Temporal.ZonedDateTime.from('2020-07-21T01:33:23[Europe/Berlin]');
zdt.withTimeZone('America/New_York'); // ⇒ 2020-07-20T19:33:23-04:00[America/New_Yo
zdt.round('hour'); // ⇒ 2020-07-21T02:00:00+02:00[Europe/Berlin]
zdt.round({ smallestUnit: 'hour', roundingMode: 'trunc' }); // ⇒ 2020-07-21T01:00:0
zdt.since(otherZonedDateTime);
zdt.until(otherZonedDateTime);
```

Manipulation

```
const date = Temporal.PlainDate.from({ year: 2024, month: 10, day: 4 });
const dateTime = date.toPlainDateTime({ hour: 22, minute: 12 });
const time = dateTime.toPlainTime();
const zonedDateTime = dateTime.toZonedDateTime('Europe/Berlin');
const instant = zonedDateTime.toInstant();
```

Special Cases

```
someInstant.add({ hours: 12 });
someInstant.add({ days: 1 });
somePlainDate.add({ hours: 24 });
zonedDateTime.until(otherZonedDateTime);
plainDateTime.until(otherplainDateTime);
Temporal.Duration.from({ minutes: 2385 }).round({ smallestUnit: 'minute', largestUni
Temporal.Duration.compare({ hours: 24 }, { days: 1 });
Temporal.Duration.compare({ years: 1 }, { days: 365 }, {relativeTo: '2024-01-01'});
Temporal.Duration.from({ days: 5, hours: 4 }).add({ years: 1 });
```

Teaser: Calendar

```
const dateTime = Temporal.ZonedDateTime.from('2020-07-01T01:00:00[Europe/Be
dateTime.until(dateTime.add({ years: 2 })); // hours: 17520
const hebrewDateTime = dateTime.withCalendar('hebrew'); // year 5780
hebrewDateTime.until(hebrewDateTime.add({ years: 2 })); // hours: 17688
```

And how is the performance?

```
for(let i=0; i<iterations; i++) {
  const instant = Temporal.Instant.from(INPUT_ISO);
  let zdt = instant.toZonedDateTimeISO(TIME_ZONE);
  zdt = zdt.add({ months: 1, days: 2, hours: 4 });
  zdt = zdt.withTimeZone("Asia/Tokyo");
  zdt.toString();
}
```

Library	Size (Min)	Size (Gzip)	Time (100k ops)
Moment	60.66 KB	19.68 KB	**1229.70 ms**
Moment-tz	776.00 KB	57.87 KB	**1606.90 ms**
Luxon	69.50 KB	21.55 KB	**3791.80 ms**
Date-fns	30.95 KB	9.35 KB	**2003.40 ms**
Dayjs	12.20 KB	4.97 KB	**17937.60 ms**
Temporal	0.38 KB	0.29 KB	**755.40 ms**
Temporal-polyfill	55.70 KB	20.57 KB	**2474.10 ms**

andrena OBJECTS

Okay, convinced. How do I switch?

- → Include polyfills at the beginning, ideally loaded only when needed
- → Switch step by step
  - → Evaluate carefully every time: `PlainDate`, `ZonedDateTime`, …
  - → Build bridges if necessary
  - → Test well! Always use realistic data for unit tests
- → Can AI help? Partially

Effort/Experience

The Project

- → Angular, Luxon
- → medium-sized; ~50k LoC (TypeScript)
- → many time/date objects (data visualization over timelines)

Result

- → Backward compatible via `temporal-polyfill`
- → ~ 135 file changes, +1400/−1000
- → 1 person, approx. 1 week, not full-time
- → 1 suboptimal part and 1 real bug identified in the old code
- → Sometimes very verbose, but helps understanding
- → Unfortunately no `endOf('day')` or similar
- → No parsing of non-ISO formats
- → Support for `<input type="date">` is still missing

andrena OBJECTS

And now your time has come!

Thank you for your attention

Marco Sieben
✉️: me@m7.codes
🌐: https://m7.codes
in: https://www.linkedin.com/in/stnimmerlein/

andrena OBJECTS