

Es wird auch langsam Zeit!

Die neue Temporal-API



Marco Sieben

- Seit bald 10 Jahren in der Softwareentwicklung bei andrena tätig
- Vorliebe für Frontend und TypeScript
- Regelmäßig auf Münchner Frontend-Meetups anzutreffen
- Hat in unterschiedlichsten Kundenprojekten immer wieder mit Zeit-/Datumsangaben zu tun gehabt

Wir haben doch schon Date



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> für mehr

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

Na, immerhin kann Date Zeitzonen

- Nicht wirklich, es zeigt nur den lokalen Offset
- Offset != Zeitzone
- Kann nicht *keinen* Offset
- Immer Datum und Zeit

Herausforderungen bei Zeit/Datum

- Exakter Zeitpunkt
- Zeit mit Zeitzone
- Nur Datum/Zeit, ohne Zone

Gut, dann nutze ich halt moment/luxon/Day.js

- **Moment.js**
 - Maintenance mode
 - mutable
 - groß und nicht Tree-Shakable
- **luxon, date-fns, Day.js**
 - Externe Abhängigkeit
 - Kein PlainDate/PlainTime

Temporal to the rescue!

- Immutable
- Typsicher
- Fail fast
- ISO-8601-konform

Gut Ding will Weile haben

- **2017:** Proposal eingebracht
- **September 2018:** Stage 1
- **Januar 2019:** Stage 2
- **März 2021:** Stage 3
- **Mai 2025:** Unterstützung in Firefox 139
- **Januar 2026:** Unterstützung in Chrome 144
- **März 2026:** Stage 4
- **Mai 2026:** Unterstützung in Node 26

Warum so lange?

- Angepasst an globale Standards (ISO 8601) und IETF extensions (RFC 9557)
- Nanosekunden statt Millisekunden
- Plain vs Civil vs Local

Aktuelle Unterstützung

Firefox	✓	seit Mai 2025
Chrome	✓	seit Januar 2026
Safari	✎	
Opera	✗	
Deno	✓	seit März 2026
Node	✓	seit Mai 2026
Bun	✗	
TypeScript	✓	seit März 2026

Für nicht unterstützende Engines gibt es Polyfills

Konzepte

- **Exakter Zeitpunkt**
 - Instant
- **Zeit mit Zeitzone**
 - ZonedDateTime
- **Einfache Zeit**
 - PlainDate
 - PlainTime
 - PlainDateTime
 - PlainMonthDay
 - PlainYearMonth
- **Zeitdifferenzen**
 - Duration

Verwendung

Now

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

Spezifische Daten

```
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);
```

Vergesst die Konstruktoren direkt wieder!

Factory-Methoden

```
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/Berlin' });  
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('de'); // 21.7.2020  
somePlainDate.toLocaleString('de', { dateStyle: 'long' }); // 21. Juli 2020  
somePlainDate.toLocaleString('de', { year: '2-digit', month: 'long' }); // Juli 20  
someZonedDateTime.toLocaleString('de'); // 21.7.2020, 23:12:12 MESZ
```

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_York]  
zdt.round('hour'); // ⇒ 2020-07-21T02:00:00+02:00[Europe/Berlin]  
zdt.round({ smallestUnit: 'hour', roundingMode: 'trunc' }); // ⇒ 2020-07-21T01:00:00+02:00[Europe/Berlin]  
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();
```


Spezialfälle

```
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', largestUnit: 'day' });
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/Berlin]');  
dateTime.until(dateTime.add({ years: 2 })); // hours: 17520  
const hebrewDateTime = dateTime.withCalendar('hebrew'); // year 5780  
hebrewDateTime.until(hebrewDateTime.add({ years: 2 })); // hours: 17688
```

Und wie ist die 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

Ok, überzeugt. Wie wechsle ich?

- Am Anfang Polyfills einbinden, idealerweise nur bei Bedarf geladen
- Stück für Stück umsteigen
 - Jedes Mal genau evaluieren: `PlainDate`, `ZonedDateTime`, ...
 - Ggf. erstmal Brücken einbauen
 - Gut testen! Bei Unit-Tests unbedingt realistische Daten verwenden
- Kann KI helfen? Teilweise

Aufwand/Erfahrung

Das Projekt

- Angular, Luxon
- mittelgroß; ~50k LoC (TypeScript)
- viele Zeit/Datumsobjekte (Datenvisualisierung über Zeitachsen)

Ergebnis

- Abwärtskompatibel via `temporal-polyfill`
- ~ 135 Dateiänderungen, +1400/-1000
- 1 Person, ca 1 Woche, nicht Vollzeit
- 1 suboptimale Stelle und **1 echten Fehler** im alten Code identifiziert
- Teilweise sehr verbose, hilft aber dem Verständnis
- Leider kein `endOf( 'day' )` o. ä.
- Kein Parsing von nicht-ISO-Formaten
- Unterstützung für `<input type="date">` fehlt noch

Und jetzt ist eure Zeit gekommen!

Danke für eure Aufmerksamkeit

Marco Sieben

✉: me@m7.codes

🌐: <https://m7.codes>

in: <https://www.linkedin.com/in/stnimmerlein/>