Timestamps for the future

A brief introduction to Temporal

Kenneth Geisshirt

https://k.zigzak.net/

Agenda

- `Date` and the packages `moment` and `datejs`
- The `Temporal` family
- Examples
- Status

`Date` and the packages `moment` ane `datejs`

- `Date` has been part of ~~Java~~ECMAScript since the beginning
  - millisecond precision
- Ken Smith ported `java.util.Date` to Brendan Eich's new language at Netscape
- The `moment` package
  - a project to fill the gaps of `Date`
  - maintainers recommend you to use `Temporal`
- The `datejs` repository was archived in July 2020

The `Temporal` family

- Nanosecond precision
- Name spaced API
  - `Temporal.Now` has static methods to give you current time
  - `Temporal.Instant` is roughly equivalent to `Date`
  - `Temporal.Duration` for time/date calculations
  - `Temporal.PlainDate` is a calendar date with or without time zone
  - `Temporal.PlainDateTime` also includes time
  - `Temporal.PlainTime` is for for time without date and time zone
  - `Temporal.ZonedDateTime` is a date/time with time zone and calendar

Parsing

- The static method `from()` is a swiss army knife for parsing date/time
- Parsing the RFC 9557 format: `YYYY-MM-DD T HH:mm:ss.sssssssss Z/±HH:mm`

```
today = Temporal.Instant.from("2026-05-07T19:00:00+02:00");
// today.toString() -> "2026-05-07T17:00:00Z"
secondToday = Temporal.Instant.from(today);
console.log(secondToday.toString());
```

Comparing

The `compare()` method is useful when sorting

```
dates = ["2026-04-01T17:24:00", "1969-07-16T09:32:00",
  "1984-04-12T07:00:00"];
launches = dates
  .map(d => Temporal.Instant.from(`${d}-04:00`))
  .sort(Temporal.Instant.compare);
// launches.map(l => l.toString()) ->
//   [ '1969-07-16T13:32:00Z', '1984-04-12T11:00:00Z',
//     '2026-04-01T21:24:00Z' ]
```

Time zone

Working with time zones is almost pleasant when using `Temporal.ZonedDataTime`

```
cphjs = Temporal.ZonedDateTime
  .from("2026-05-07T18:00:00+02:00[Europe/Copenhagen]");
isLeapYear = cphjs.inLeapYear; // false

transition = cphjs.getTimeZoneTransition("next");
// 2026-10-25T02:00:00+01:00[Europe/Copenhagen]
transition2 = transition.getTimeZoneTransition("next");
// 2027-03-28T03:00:00+02:00[Europe/Copenhagen]

inIceland = cphjs.withTimeZone("Atlantic/Reykjavik");
// 2026-05-07T16:00:00+00:00[Atlantic/Reykjavik]
transition3 = inIceland.getTimeZoneTransition("next");
// null - lucky bastards!
```

Next week

`Temporal.Duration` is powerful for date calculations

```
now = Temporal.Now.plainDateISO();
oneWeek = Temporal.Duration.from({ weeks: 1 });
nextWeek = now.add(oneWeek);
```

`Date` is a bit brittle

```
now = new Date();
oneWeek = 1000 * 60 * 60 * 24 * 7;
nextWeek = new Date(now.getTime() + oneWeek);
```

Until

`Temporal` is very direct:

```
now = Temporal.Now.instant();
eclipse = Temporal.Instant.from("2026-08-12T20:26+02:00");
daysUtilEclipse = now.until(eclipse).total("days");
```

With `Date` you go through milliseconds

```
now = new Date();
eclipse = new Date("2026-08-12T20:26+02:00");
msUntil = eclipse.getTime() - now.getTime();
daysUntilEclipse = msUntil / (1000 * 60 * 60 * 24);
```

Calendar

```
date = Temporal.PlainDate.from("2026-05-07");
// date.toLocaleString("en-US") -> "5/7/2026"
// date.toLocaleString("da-DA") -> "7.5.2026"

japanese = date.withCalendar("japanese");
// japanese.toLocaleString("jp-JP",
//   { calendar: "japanese" }) -> "5/7/8 R"
islamic = date.withCalendar("islamic-umalqura");
// islamic.toLocaleString("en-US",
//   { calendar: "islamic-umalqura" }) -> "11/20/1447 AH"
```

Status

- Works in Chrome (144) and Firefox (139)
  - and for Safari it must explicitly be enabled by the user
- Deno and Kiesel have support
  - Bun is still in progress
- Node 26 (released May 5, 2026)
  - use polyfills for older versions
- `Temporal` was added to TypeScript in version 6.0 (released March 23, 2026)

Summary

- The `Temporal` family is ready for the party
- If you are using `moment` and/or `datejs`, you should consider to migrate
- Go through your code bases to identify your usage of `Date` - and replace it