JavaScript Dates: Beginner to Advanced Guide

Overview

JavaScript Dates are powerful but unintuitive. This guide teaches how they really work, why bugs happen, and how to avoid them.

---

1. The Zero-Indexed Trap

JavaScript mixes 0-based and 1-based values.

Method	Meaning	Range
getMonth()	Month	0–11
getDate()	Day of month	1–31
getDay()	Day of week	0–6

Rule of Thumb

- 0 means **January** or **Sunday**
- If you want "the 19th", use `getDate()`

```
const date = new Date('2026-10-19');
date.getMonth(); // 9
date.getDate();  // 19
date.getDay();   // 1



2. Creating Dates Correctly
Numeric Constructor
new Date(2026, 0, 1); // Jan 1, 2026

ISO Strings (Recommended)
new Date('2026-10-31');


⚠ Avoid locale strings like:

new Date('10/31/2026');

3. The Unix Epoch

All JS dates are stored as:

Milliseconds since Jan 1, 1970 (UTC)

```

```
Date.now();

Comparing Dates
dateA.getTime() > dateB.getTime();

4. Date Mutation (Critical Concept)

Dates are mutable.

const d = new Date();
d.setMonth(11); // changes original

Safe Pattern
const copy = new Date(d);
copy.setMonth(11);

5. Timezones

ISO dates default to UTC.

new Date('2026-10-31');


May appear as previous day in some regions.

Local-Safe Creation
new Date(2026, 9, 31);

6. Formatting Dates
Use toLocaleString
date.toLocaleString('en-US', {
  weekday: 'long',
  year: 'numeric',
  month: 'long',
  day: 'numeric'
});

7. Date Math
function addDays(date, days) {
  const d = new Date(date);
  d.setDate(d.getDate() + days);
  return d;
}
```