perplexity

Why the Millisecond Approach for Dates in JavaScript?

JavaScript represents dates and times using the **number of milliseconds since midnight, January 1, 1970 UTC** (known as the Unix epoch).[1] [2] [3]

Reasons for Using Milliseconds

- **Precision**: Milliseconds provide enough granularity for almost all real-world time-tracking use cases, allowing calculations down to 1/1000th of a second.[2] [3]
- **Simplicity in Arithmetic**: Storing dates as a single integer makes operations like difference, comparison, and addition straightforward. For example, subtracting two timestamps directly gives the duration in milliseconds.[4] [3]
- **Time Zone Independence**: Millisecond timestamps are always in UTC, making them *timezone-agnostic* and consistent across systems and platforms.[3] [1]
- **Interoperability**: Numeric timestamps are simple to serialize, store, and send between systems—most programming languages use a similar approach, making cross-language data exchange easier.[5]
- **Performance**: Numeric representation is more efficient to store, manipulate, and compare than complex objects or formatted date strings. Functions like `Date.now()` or `getTime()` return these numbers instantly without extra object overhead.[4] [3]
- **Consistent Storage**: Millisecond values are not affected by varying string formats, daylight saving adjustments, or locale differences.

Practical Uses

- **Measure elapsed time**:
  ```
  const start = Date.now();
  // ... some code ...
  const end = Date.now();
  const elapsed = end - start; // in milliseconds
  ```
- **Store and compare dates**: Milliseconds are ideal for fast comparisons or storing exact times in databases/logs.[3]
- **Cross-system compatibility**: As most systems can parse and manipulate a Unix epoch timestamp, this approach ensures compatibility.[5]

In summary, the millisecond approach for date storage and manipulation in JavaScript is chosen for its **precision, simplicity, and universality**—making date handling robust and consistent across environments.[2] [4] [3]

⁂

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date
2. https://www.w3schools.com/js/js_dates.asp
3. https://mimo.org/glossary/javascript/date-gettime
4. https://javascript.info/date
5. https://stackoverflow.com/questions/47456537/should-i-use-date-objects-or-integer-timestamps-for-handling-dates-in-javascript