Cron Job Libraries Comparison

`node-cron` vs `croner`

Both `node-cron` and `croner` are used to schedule **cron jobs** in Node.js applications.

Feature Comparison

Feature	`node-cron`	`croner`
Usage Purpose	Scheduling recurring tasks	Scheduling recurring or one-time tasks
Cron Syntax Support	Basic (5-6 fields)	Advanced (5-7 fields, `L`, `#`, ISO strings)
Time Zone Support	Not supported	Yes (e.g., `timezone: 'Asia/Kolkata'`)
Pause / Resume	No	Yes (`pause()`, `resume()`)
Manual Triggering	Not available	Yes (`trigger()`)
One-Time Date Schedules	No	Yes (`new Cron('2025-08-12T08:00:00')`)
Get Next Run Times	No	Yes (`job.nextRuns(5)`)
Built-in Error Handling / Overrun	No	Yes
Cross-Platform (Deno, Bun, Browser)	Node.js only	Fully supported
Dependencies	0	0
Best For	Simple cron jobs	Advanced, reliable, and timezone-aware scheduling

NODE-CRON.js

```js
// ✅ Usage Purpose: Recurring Task
const nodeTask = cron.schedule('*/5 * * * * *', () => {
  console.log(`[node-cron] ✅ Triggered at ${new Date().toLocaleTimeString()}`);
});

//  Pause / Resume: Not supported
//  Manual Trigger: Not supported
// Timezone: Not supported
// One-Time Date Schedules: Not supported
//  get  Next Runs: Not supported
//  Error Handling: Not built-in

Tabnine | Edit | Test | Explain | Document
setTimeout(() => {
```

```
PS C:\Users\Administrator\OneDrive\Desktop\new> node --version
v20.19.0
PS C:\Users\Administrator\OneDrive\Desktop\new> node node-cron.js
[node-cron] ✅ Triggered at 12:15:35 PM
[node-cron] ✅ Triggered at 12:15:40 PM
[node-cron] ✅ Triggered at 12:15:45 PM
[node-cron] ⛔ Stopped
PS C:\Users\Administrator\OneDrive\Desktop\new>
```

`cron.schedule(...)`: schedules a job every 5 seconds (`*/5 * * * * *`)

It **logs a message** every time it runs

After 20 seconds, it **stops** the job with `task.stop()`

Feature	Demonstrated?	How?
Recurring Schedule	Yes	Cron syntax every 5 sec
Pause / Resume	No	Not supported
Manual Trigger	No	Not supported
Time Zone	No	Runs in local server time
One-Time Job	No	Not supported
View Next Run Times	No	Not supported

Croner.js

```
croner.js > ...
const { Cron } = require('croner');

const job = new Cron('*/5 * * * * *', { timezone: 'Asia/Kolkata' }, (scheduled) => {
  console.log(`[croner] ✅ Triggered (scheduled: ${scheduled})`);
});

// Pause after 15 sec
Tabnine | Edit | Test | Explain | Document
setTimeout(() => {
  job.pause();
  console.log('[croner] ⏸️ Paused');
```

```
PRO  Focus folder in explorer (ctrl + click)   TERMINAL   PORTS                    Code +
PS C:\Users\Administrator\OneDrive\Desktop\new> node "c:\Users\Administrator\OneDrive\Desktop\new\croner.js"
Next 3 runs: [
  2025-07-10T06:53:15.000Z,
  2025-07-10T06:53:15.000Z,
  2025-07-10T06:53:20.000Z,
  2025-07-10T06:53:25.000Z
]
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ⏸️ Paused
[croner] ▶️ Resumed
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ✅ Triggered (scheduled: [object Object])
[croner] 🔁 Manually triggered
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ✅ Triggered (scheduled: [object Object])
[croner] ⛔ Stopped
PS C:\Users\Administrator\OneDrive\Desktop\new>
```

`new Cron(...)`: schedules a job every 5 seconds with **timezone support**

It logs the **scheduled run time**

After 15 seconds: pauses the job

After 25 seconds: resumes the job

After 30 seconds: manually triggers the job (even if not scheduled)

After 40 seconds: fully stops the job

Also prints **next 3 upcoming run times**

Feature	Demonstrated?	How?
Recurring Schedule	Yes	Cron syntax every 5 sec
Pause / Resume	Yes	`job.pause()` / `job.resume()`
Manual Trigger	Yes	`job.trigger()`
Time Zone	Yes	`{ timezone: 'Asia/Kolkata' }`
One-Time Job	Yes (commented)	`new Cron('2025-08-12T08:00:00')`
View Next Run Times	Yes	`job.nextRuns(3)`

COMPARISON

Test Element	Shown in Code	node-cron	croner
Basic Scheduling	Runs every 5 seconds	Supported	Supported
Stop Job	`task.stop()` / `job.stop()`	Supported	Supported
Pause & Resume	`pause()`, `resume()` (only croner)	Not available	Demonstrated
Manual Trigger	`trigger()`	Not available	Demonstrated
Timezone Support	Asia/Kolkata	No	Yes
View Next Runs	`job.nextRuns(3)`	No	Yes
One-Time Job	Optional in Croner	No	Yes (commented)

Issues

several users have reported **high memory/CPU usage** and eventual **crashes when using `node-cron` with PM2** for long-running tasks.
Using **node-cron 3.0.1** on **Node.js v16**.Over time, the app **eats up memory** and eventually crashes.The cron jobs are **asynchronous**, and using **PM2** to keep the script alive. Several users on Reddit and other forums report **node-cron jobs not firing consistently** in production—especially in environments that pause or sleep when idle .

CAUSES

1. **Memory Leak in node-cron** (especially in long-running apps).

2. **Unawaited async functions** inside cron jobs.

3. **PM2 + node-cron conflict**: PM2 tries to restart on crash, creating an infinite loop.

4. **Too many event listeners** accumulating over time.

Solution

1) **Use `croner` Instead (Alternative Scheduler)**
2) `node-cron` is known to leak memory in some use cases. **Croner** is a **drop-in replacement**, more modern and stable.
3) Some solutions include switching to system cron/jobs (e.g., native `cron`, cloud scheduler) or using `croner` as an alternative due to reliability

Library	Strengths	Main Issues	Suggestions
node-cron	Lightweight, familiar cron syntax	- Memory leaks on Node 18+ - Incorrect cron patterns - Instability if runtime becomes idle	- Add seconds param - Upgrade Node - Use workaround PR code - Use defensive error-handling - Avoid host platforms that sleep
croner	Modern, zero-deps, Deno/Bun support	- Lack of immediate execution with interval syntax - Missing day-offset scheduling	- Ideal if immediate firing and day-offset support are not needed - Otherwise, fork and implement feature or wait for update

CONCLUSION

- Choose `node-cron` if:
  - We need simple recurring tasks.
  - We want a lightweight, minimal dependency scheduler.
- Choose `croner` if:
  - We need **timezone-aware jobs**.
  - We want the ability to **pause, resume, or trigger manually**.
  - We want to prevent **overlaps or errors** in job execution.
  - We are scheduling jobs for **future dates or exact timestamps**.