readme.md for @push.rocks/smarttime

Handle time in smart ways — cron scheduling, extended dates, precise measurements, timers, intervals, and human-readable formatting. □□

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Install

```
npm install @push.rocks/smarttime
```

or with pnpm:

```
pnpm install @push.rocks/smarttime
```

Usage

`@push.rocks/smarttime` is a comprehensive TypeScript-first toolkit for working with time. It covers everything from cron-based scheduling and extended date manipulation to high-resolution measurements, timers, intervals, and human-readable time formatting.

All examples below use ESM imports and TypeScript.

□ Time Units & Calculations

Convert human-readable time durations into milliseconds using the `units` helpers and `getMilliSecondsFromUnits`:

```
import { units, getMilliSecondsFromUnits } from '@push.rocks/smarttime';

// Individual unit conversions
const oneDay = units.days(1);       // 86400000
const twoHours = units.hours(2);    // 7200000
const thirtySeconds = units.seconds(30); // 30000

// Combined duration — great for timeouts, TTLs, cache expiration, etc.
const duration = getMilliSecondsFromUnits({
  years: 1,
  months: 2,
  weeks: 3,
  days: 4,
  hours: 5,
  minutes: 6,
  seconds: 7,
});

console.log(`Duration: ${duration}ms`);
```

□□ Human-Readable Time Formatting

Turn raw milliseconds into a friendly string, or get a relative "time ago" description:

```
import {
  getMilliSecondsAsHumanReadableString,
  getMilliSecondsAsHumanReadableAgoTime,
} from '@push.rocks/smarttime';

// "2h 30m"
console.log(getMilliSecondsAsHumanReadableString(9_000_000));

// "5 minutes ago" style output
const fiveMinutesAgo = Date.now() - 5 * 60 * 1000;
console.log(getMilliSecondsAsHumanReadableAgoTime(fiveMinutesAgo));
```

□□ CronManager — Scheduled Task Execution

The `CronManager` lets you register and run multiple cron jobs using standard 6-field cron syntax (seconds included). Powered by croner.

```
import { CronManager } from '@push.rocks/smarttime';

const cronManager = new CronManager();

// Run every 10 seconds
const job1 = cronManager.addCronjob('*/10 * * * * *', async (triggerTime) => {
  console.log(`Job 1 triggered at ${new Date(triggerTime).toISOString()}`);
});

// Run every full minute
const job2 = cronManager.addCronjob('0 * * * * *', async () => {
  console.log('Full minute tick!');
});

// Start all registered jobs
cronManager.start();

// Later: remove a specific job
cronManager.removeCronjob(job1);

// Stop all jobs
cronManager.stop();
```

The `CronManager` automatically calculates the shortest sleep interval between jobs and efficiently schedules wake-ups — no polling required.

□□ ExtendedDate — Enhanced Date Handling

`ExtendedDate` extends the native JavaScript `Date` with factory methods for common date formats and useful inspection methods:

```
import { ExtendedDate } from '@push.rocks/smarttime';

// From European format: "DD.MM.YYYY"
const date1 = ExtendedDate.fromEuropeanDate('25.12.2024');

// From European date + time with timezone context
const date2 = ExtendedDate.fromEuropeanDateAndTime('25.12.2024', '14:30:00', 'Europe/Berlin');

// From hyphenated date: "YYYY-MM-DD"
const date3 = ExtendedDate.fromHyphedDate('2024-12-25');

// From milliseconds
const date4 = ExtendedDate.fromMillis(Date.now());

// From an existing Date object
const date5 = ExtendedDate.fromDate(new Date());

// Export back to various formats
console.log(date1.exportToEuropeanDate());         // "25.12.2024"
console.log(date1.exportToHyphedSortableDate());   // "2024-12-25"

// Get structured date units
const units = date1.exportToUnits();
console.log(units.monthName);         // "December"
console.log(units.dayOfTheWeekName);  // "Wednesday"

// Custom formatting (via dayjs)
console.log(date1.format('YYYY-MM-DD HH:mm'));

// Boolean checks
console.log(date4.isToday());  // true

// Time comparison: has less than 1 hour passed since this date?
console.log(date4.lessTimePassedToNow({ hours: 1 }));  // true
console.log(date4.moreTimePassedToNow({ days: 1 }));   // false
```

⏱️ HrtMeasurement — High-Resolution Timing

Measure the duration of operations with precision:

```
import { HrtMeasurement } from '@push.rocks/smarttime';

const measurement = new HrtMeasurement();
measurement.start();

// ... perform some operation ...

measurement.stop();
console.log(`Took ${measurement.milliSeconds}ms`);
console.log(`Took ${measurement.nanoSeconds}ns`);

// Reset and reuse
measurement.reset();
measurement.start();
// ...
```

⏳ Timer — Pausable Countdown

A promise-based timer with pause, resume, and reset capabilities:

```
import { Timer } from '@push.rocks/smarttime';

const timer = new Timer(5000); // 5-second countdown
timer.start();

// Await completion
await timer.completed;
console.log('Timer finished!');

// Or use pause/resume
const timer2 = new Timer(10000);
timer2.start();

// Pause after 3 seconds
setTimeout(() => timer2.pause(), 3000);

// Resume later
```

```
setTimeout(() => timer2.resume(), 6000);

await timer2.completed;

// Reset completely and start over
timer2.reset();
timer2.start();
```

Check remaining time with `timer.timeLeft`.

□□ Interval — Repeating Jobs

Run functions at a fixed interval with start/stop control:

```
import { Interval } from '@push.rocks/smarttime';

const interval = new Interval(2000); // Every 2 seconds

interval.addIntervalJob(() => {
  console.log('Tick!', new Date().toISOString());
});

interval.addIntervalJob(() => {
  console.log('Another parallel job');
});

interval.start();

// Stop later
setTimeout(() => interval.stop(), 10000);
```

□□ TimeStamp — Comparison & Tracking

The `TimeStamp` class captures a moment in time and provides comparison utilities:

```
import { TimeStamp } from '@push.rocks/smarttime';

const stamp1 = new TimeStamp();
```

```
// ... some time passes ...
const stamp2 = new TimeStamp();

console.log(stamp1.milliSeconds);  // Unix ms
console.log(stamp1.epochtime);     // Unix seconds

// Comparison
console.log(stamp1.isOlderThan(stamp2));                  // true
console.log(stamp2.isYoungerThanOtherTimeStamp(stamp1));   // true

// Derived timestamps track change
const derived = TimeStamp.fromTimeStamp(stamp1);
console.log(`${derived.change}ms have passed since stamp1`);

// From known milliseconds
const known = TimeStamp.fromMilliSeconds(1700000000000);
```

API Overview

Export	Description
`CronManager`	Register and run multiple cron jobs with automatic scheduling
`CronJob`	Individual cron job (created via `CronManager.addCronjob()`)
`ExtendedDate`	Enhanced `Date` with factory methods and formatting
`HrtMeasurement`	High-resolution time measurement
`Timer`	Promise-based countdown with pause/resume/reset
`Interval`	Repeating job execution at fixed intervals
`TimeStamp`	Point-in-time capture with comparison utilities
`units`	Time unit conversion functions (`days()`, `hours()`, etc.)
`getMilliSecondsFromUnits()`	Convert combined time units to milliseconds
`getMilliSecondsAsHumanReadableString()`	Format ms as human-readable string
`getMilliSecondsAsHumanReadableAgoTime()`	Format timestamp as relative "ago" string

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH
Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:12:38 UTC by foss.global Team
Updated 2026-03-28 12:19:24 UTC by foss.global Team