@push.rocks/smartwatch

A smart watch module for the TypeScript space.

- readme.md for @push.rocks/smartwatch
- changelog.md for @push.rocks/smartwatch

readme.md for @push.rocks/smartwatch

A cross-runtime file watcher with glob pattern support for Node.js, Deno, and Bun.

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/smartwatch
# or
pnpm add @push.rocks/smartwatch
```

Features

- **Cross-Runtime** — Works on Node.js 20+, Deno, and Bun
- **Glob Pattern Support** — Watch files using patterns like `**/*.ts` and `src/**/*.{js,jsx}`
- **RxJS Observables** — Subscribe to file system events using reactive streams
- **Dynamic Watching** — Add or remove watch patterns at runtime
- **Write Stabilization** — Built-in debouncing and awaitWriteFinish support for atomic writes
- **TypeScript First** — Full TypeScript support with comprehensive type definitions

How It Works

smartwatch selects the best file watching backend for the current runtime:

Runtime	Backend
Node.js/Bun	chokidar v5 (uses `fs.watch()` internally)
Deno	Native `Deno.watchFs()` API

On Node.js and Bun, chokidar provides robust cross-platform file watching with features like atomic write detection, inode tracking, and write stabilization. On Deno, native APIs are used directly with built-in debouncing and temporary file filtering.

Glob patterns are handled through picomatch — base paths are extracted from patterns and watched natively, while events are filtered through matchers before emission.

Usage

Basic Setup

```
import { Smartwatch } from '@push.rocks/smartwatch';

// Create a watcher with glob patterns
const watcher = new Smartwatch([
  './src/**/*.ts',
  './public/assets/**/*'
]);

// Start watching
await watcher.start();
```

Subscribing to File Events

Use RxJS observables to react to file system changes:

```
// Get an observable for file changes
const changeObservable = await watcher.getObservableFor('change');
changeObservable.subscribe(([path, stats]) => {
  console.log(`File changed: ${path}`);
```

```
  console.log(`New size: ${stats?.size} bytes`);
});

// Watch for new files
const addObservable = await watcher.getObservableFor('add');
addObservable.subscribe(([path]) => {
  console.log(`File added: ${path}`);
});

// Watch for deleted files
const unlinkObservable = await watcher.getObservableFor('unlink');
unlinkObservable.subscribe(([path]) => {
  console.log(`File deleted: ${path}`);
});
```

Supported Events

Event	Description
`add`	File has been added
`addDir`	Directory has been added
`change`	File has been modified
`unlink`	File has been removed
`unlinkDir`	Directory has been removed
`error`	Error occurred
`ready`	Initial scan complete

Dynamic Watch Management

Add or remove patterns while the watcher is running:

```
const watcher = new Smartwatch(['./src/**/*.ts']);
await watcher.start();

// Add more patterns to watch
watcher.add(['./tests/**/*.spec.ts', './config/*.json']);
```

```
// Remove a pattern
watcher.remove('./src/**/*.test.ts');
```

Stopping the Watcher

```
await watcher.stop();
```

Complete Example

```
import { Smartwatch } from '@push.rocks/smartwatch';

async function watchProject() {
  const watcher = new Smartwatch([
    './src/**/*.ts',
    './package.json'
  ]);

  await watcher.start();
  console.log('Watching for changes...');

  const changes = await watcher.getObservableFor('change');
  changes.subscribe(([path, stats]) => {
    console.log(`Modified: ${path} (${stats?.size ?? 'unknown'} bytes)`);
  });

  const additions = await watcher.getObservableFor('add');
  additions.subscribe(([path]) => {
    console.log(`New file: ${path}`);
  });

  const deletions = await watcher.getObservableFor('unlink');
  deletions.subscribe(([path]) => {
    console.log(`Deleted: ${path}`);
  });

  // Handle graceful shutdown
  process.on('SIGINT', async () => {
```

```
    await watcher.stop();
    process.exit(0);
  });
}

watchProject();
```

API Reference

`Smartwatch`

Constructor

```
new Smartwatch(patterns: string[])
```

Creates a new Smartwatch instance with the given glob patterns.

Parameters:

- `patterns` — Array of glob patterns to watch (e.g., `['./src/**/*.ts', './config/*.json']`)

Methods

Method	Returns	Description
`start()`	`Promise<void>`	Starts watching for file changes
`stop()`	`Promise<void>`	Stops the file watcher and cleans up resources
`add(patterns: string[])`	`void`	Adds additional patterns to watch
`remove(pattern: string)`	`void`	Removes a pattern from the watch list
`getObservableFor(event: TFsEvent)`	`Promise<Observable<[string, Stats]>>`	Returns an RxJS observable for the specified event

Properties

Property	Type	Description
`status`	`'idle' | 'starting' | 'watching'`	Current watcher status

Types

```
type TFsEvent = 'add' | 'addDir' | 'change' | 'error' | 'unlink' | 'unlinkDir' | 'ready' |
'raw';
type TSmartwatchStatus = 'idle' | 'starting' | 'watching';
```

Requirements

Runtime	Version
Node.js	20+
Deno	Any version with `Deno.watchFs()` support
Bun	Uses Node.js compatibility layer

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.

changelog.md for @push.rocks/smartwatch

2026-03-23 - 6.4.0 - feat(watchers)

add Rust-powered watcher backend with runtime fallback and cross-platform test coverage

- introduces a new Rust watcher binary and TypeScript bridge using @push.rocks/smartrust
- updates watcher selection to prefer the Rust backend when available and fall back to Node.js or Deno implementations
- improves Deno event classification for any/other file system events
- prevents Node.js watcher shutdown from affecting unrelated FSWatcher handles
- adds platform-specific tests for Node.js, Deno, Bun, and Rust-backed watchers

2026-03-23 - 6.3.1 - fix(watcher)

unref lingering FSWatcher handles after stopping the node watcher

- Ensures chokidar file watcher handles do not keep the process running after watcher shutdown
- Works around chokidar v5 behavior where close() can resolve before all fs.watch() handles are fully released

2025-12-11 - 6.3.0 - feat(watchers)

Integrate chokidar-based Node watcher, expose awaitWriteFinish options, and update docs/tests

- Add chokidar dependency and implement NodeWatcher as a chokidar wrapper for Node.js/Bun
- Expose awaitWriteFinish, stabilityThreshold and pollInterval in IWatcherOptions and wire them into the NodeWatcher
- Update watcher factory to return NodeWatcher for Node/Bun and DenoWatcher for Deno

- Adjust tests to wait for chokidar readiness and to expect chokidar's atomic handling (delete+recreate -> change)
- Revise README and technical hints to document chokidar usage and cross-runtime behavior

2025-12-11 - 6.2.5 - fix(watcher.node)

Normalize paths and improve Node watcher robustness: restart/rescan on errors (including ENOSPC), clear stale state, and remove legacy throttler

- Normalize all paths to absolute at watcher entry points (watchPath, handleFsEvent, scanDirectory) to avoid relative/absolute mismatch bugs
- On watcher restart: clear pending unlink timeouts, dispose stale DirEntry data, and perform a rescan to catch files created during the restart window
- Trigger watcher restart on ENOSPC (inotify limit) errors instead of only logging the error
- Remove the previous Throttler implementation and rely on the existing debounce + event-sequence tracking to handle rapid events
- Atomic write handling and queued unlink behavior preserved; pending unlinks are cleared for restarted base paths to avoid stale events

2025-12-11 - 6.2.4 - fix(tests)

Stabilize tests and document chokidar-inspired Node watcher architecture

- test: add waitForFileEvent helper to wait for events for a specific file (reduces test flakiness)
- test: add small delays after unlink cleanup to account for atomic/temp-file debounce windows
- docs: expand readme.hints.md with a detailed Node watcher architecture section (DirEntry, Throttler, atomic write handling, closer registry, constants and config)
- docs: list updated test files and coverage scenarios (inode detection, atomic writes, stress tests)

2025-12-11 - 6.2.3 - fix(watcher.node)

Improve handling of temporary files from atomic editor writes in Node watcher

- Detect temporary files produced by atomic editor saves and attempt to map them to the real target file instead of silently skipping the event
- Add getTempFileTarget() to extract the real file path from temp filenames (supports patterns like file.ts.tmp.PID.TIMESTAMP and generic .tmp.*)
- When a temp-file event is seen, queue a corresponding event for the resolved real file after a short delay (50ms) to allow rename/replace to complete
- Add logging around temp file detection and real-file checks to aid debugging

2025-12-11 - 6.2.2 - fix(watcher.node)

Defer events during initial scan, track full event sequences, and harden watcher shutdown

- Defer fs.watch events that arrive during the initial directory scan and process them after the scan completes to avoid race conditions where watchedFiles isn't populated.
- Debounce now tracks the full sequence of events per file (rename/change) instead of collapsing to the last event, preventing intermediate events from being lost.
- Detect delete+recreate via inode changes and emit unlink then add when appropriate; handle rapid create+delete sequences by emitting both events.
- During stop(), cancel pending debounced emits before flipping _isWatching and make handleFsEvent return early when watcher is stopped to prevent orphaned timeouts and post-stop emits.
- Add verbose logging of event sequences to aid debugging of complex fs event scenarios.
- Update tests to expect unlink + add for inode replacement scenarios.
- Version bump from 6.2.1 → 6.2.2

2025-12-10 - 6.2.1 - fix(watcher.node)

Handle fs.watch close without spurious restarts; add tests and improve test runner

- Prevent spurious restarts and noisy warnings on fs.watch 'close' by checking the internal isWatching flag before logging and restarting (ts/watchers/watcher.node.ts).
- Add comprehensive test suites covering basic operations, inode-change detection, atomic writes and stress scenarios (test/test.basic.ts, test/test.inode.ts, test/test.stress.ts).

- Remove outdated test (test/test.ts) and delete the test asset test/assets/hi.txt.
- Update test script in package.json to enable verbose logging, write a logfile and increase timeout to 120s to reduce flakiness in test runs.

2025-12-10 - 6.2.0 - feat(watchers)

Improve Node watcher robustness: file-level inode tracking, abortable restarts, restart race guards, and untracked-file handling

- Add file-level inode tracking to detect delete+recreate (editor atomic saves) and emit correct 'change'/'add' events
- Make restart delays abortable via AbortController so stop() cancels pending restarts and prevents orphan watchers
- Guard against concurrent/dual restarts with restartingPaths to avoid race conditions between health checks and error handlers
- Emit 'unlink' for deletions of previously untracked files (files created after initial scan) and clean up inode state
- Track file inodes during initial scan and update/cleanup inode state on events
- Improve logging for restart/inode/delete+recreate scenarios and update documentation/readme hints to v6.2.0+

2025-12-08 - 6.1.1 - fix(watchers/watcher.node)

Improve Node watcher robustness: inode tracking, ENOSPC detection, enhanced health checks and temp-file handling

- Track directory inodes (watchedInodes) and restart watchers if inode changes are detected (addresses stale watchers when directories are replaced).
- Health check now validates inode stability and explicitly detects ENOSPC (inotify max_user_watches) errors, emitting errors and logging a recommended fix command.
- Detect ENOSPC in FSWatcher error events and log guidance to increase inotify limits.
- Clear inode tracking state on watcher stop to avoid stale state across restarts.
- Improve temporary file handling and logging to avoid dropping events for atomic writes (only skip pure temp files and log skipped temp events).
- Documentation (readme.hints.md) updated with robustness notes, known fs.watch limitations, and example logs.

2025-12-08 - 6.1.0 - feat(watcher.node)

Add automatic restart, periodic health checks, and safe event emission to Node watcher; improve logging and stat handling

- NodeWatcher: introduced safeEmit() to isolate subscriber errors and prevent watcher crashes
- Auto-restart on failure with exponential backoff (1s → 30s) and up to 3 retry attempts per watched base path
- Periodic health checks (every 30s) to detect missing watched paths and trigger automatic restarts
- Handle unexpected FSWatcher 'close' events and restart watchers when they close silently
- Verbose lifecycle logging with `[smartwatch]` prefix for start/stop/health/restart events
- Clear restart tracking and stop health checks on watcher.stop() to ensure clean shutdown
- Improved statSafe() to normalize followSymlinks logic and log non-ENO errors as warnings
- Updated readme.hints.md documenting the new robustness features (v6.1.0+)

2025-12-08 - 6.0.0 - BREAKING CHANGE(watchers)

Replace polling-based write stabilization with debounce-based event coalescing and simplify watcher options

- Remove polling-based WriteStabilizer (ts/utils/write-stabilizer.ts) and related waitForWriteFinish logic
- Introduce debounce-based coalescing (debounceMs) for Node and Deno watchers (ts/watchers/watcher.node.ts, ts/watchers/watcher.deno.ts)
- Change IWatcherOptions: remove stabilityThreshold/pollInterval/maxWaitTime and add debounceMs
- Default watcher options updated to use debounceMs = 100
- Node/Deno watchers now skip editor temp files earlier, debounce duplicate events, and emit events directly (no size polling)
- Updated default watcher creation in Smartwatch to pass debounceMs
- Update package.json build script to run 'tsbuild tsfolders'

2025-12-08 - 5.1.0 - feat(watchers)

Improve write stabilization and ignore temporary editor files

- Add maxWaitTime option (ms) to IWatcherOptions and WriteStabilizer to cap how long stabilization will wait (default: 1000ms).
- WriteStabilizer: reduce default stabilityThreshold from 300ms to 100ms and track write start time to enforce maxWaitTime and avoid indefinite polling.
- Node and Deno watchers: detect and ignore common temporary/editor files (e.g. *.tmp.*, *.swp, *.swx, trailing ~, .#*) to prevent spurious events from atomic saves.
- Node watcher: treat rename (atomic-save) events as already-complete files and emit add/change immediately without stabilization.
- Deno watcher: use the configured maxWaitTime and polling-based stabilization for modify events to ensure consistent behavior across runtimes.

2025-11-30 - 5.0.0 - BREAKING CHANGE(@push.rocks/smartwatch)

Rename package and update branding/docs: switch from @push.rocks/smartchok to @push.rocks/smartwatch, update repository/homepage/bugs URLs and author, and refresh README examples and install instructions.

- Package name changed from @push.rocks/smartchok to @push.rocks/smartwatch in package.json
- Repository, homepage and issue URLs updated to point to the new smartwatch repository
- Author changed to Task Venture Capital GmbH in package metadata
- README updated: install commands, import examples and references now use @push.rocks/smartwatch
- Documentation text and branding updated throughout README (project name, internal references)

2025-11-30 - 4.0.1 - fix(readme)

Update README: refine description and clarify trademark/legal information

- Refined project description and tagline for clarity and brevity (adds lightweight wording and an emoji).

- Updated Trademark section to explicitly mention third-party trademarks, add guidance about usage and approval, and clarify that trademarks are not covered by the MIT license.
- Minor legal/company wording and formatting fixes (e.g. 'District Court' capitalization and contact sentence tweaks).
- General README wording and formatting improvements for clearer instructions and feature descriptions.

2025-11-30 - 4.0.0 - BREAKING CHANGE(watchers)

Replace chokidar with native platform watchers and add cross-runtime support (Node.js, Deno, Bun); introduce write stabilization and internal glob matching

- Replaced chokidar-based implementation with native file watching APIs (Node.js fs.watch, Deno.watchFs).
- Added platform-specific watchers: NodeWatcher and DenoWatcher (Bun uses Node compatibility).
- Implemented polling-based write stabilization (awaitWriteFinish replacement) to avoid duplicate events during writes.
- Keep glob pattern support by matching events internally using picomatch; base-path extraction used to limit watch scope.
- API/runtime requirement increased: Node.js >= 20.0.0 is required for native recursive fs.watch.
- Package/documentation name and examples updated to @push.rocks/smartchok and export the Smartwatch class.

2025-11-30 - 3.0.0 - BREAKING CHANGE(smartwatch)

Introduce Smartwatch: cross-runtime native file watching for Node.js, Deno and Bun; rename smartchok to smartwatch and bump major version to 2.0.0

- Rename public API and docs from Smartchok to Smartwatch and update package metadata for the new module name.
- Replace chokidar with native watchers and picomatch-based glob filtering to enable cross-runtime support (Node.js, Deno, Bun).

- Add watcher factory and runtime-specific implementations: watchers/index.ts, watcher.node.ts, watcher.deno.ts.
- Add WriteStabilizer (ts/utils/write-stabilizer.ts) to provide awaitWriteFinish functionality via polling.
- Introduce @push.rocks/smartenv for runtime detection and remove direct chokidar dependency; update dependencies accordingly.
- Update tests (test/test.ts) and documentation (readme.md, readme.hints.md) to reflect API/name changes and new architecture.
- Bump package version to 2.0.0 to mark breaking changes in API and behavior.

2025-11-29 - 1.2.0 - feat(core)

Migrate to chokidar 5.x, add picomatch filtering and update test/dev dependencies

- Upgrade runtime dependencies: chokidar -> ^5.0.0 and picomatch -> ^4.0.3; bumped related @push.rocks packages versions.
- Upgrade devDependencies: @git.zone/tsbuild, @git.zone/tsrun and @git.zone/tstest to newer v2/v3 releases; updated @types/node.
- Updated README and readme.hints to document migration to chokidar 5.x and dev dependency changes.
- Tests updated to use @git.zone/tstest/tapbundle (import change) and test runner start changed to export default tap.start().
- Smartchok implementation updated to extract glob base paths, watch base directories and filter events via picomatch matchers (shouldWatchPath + event filtering).
- Note: ts/00_commitinfo_data.ts still references chokidar 4.x in the description and should be updated to reflect the migration.

2025-06-26 - 1.1.1 - fix(package.json)

Add packageManager field to package.json for pnpm configuration

- Added "packageManager": "pnpm@10.11.0+sha512.6540583f41cc5f628eb3d9773ecee802f4f9ef9923cc45b69890fb47991d4b092964694ec3a4f738a420c918a333062c8b925d312f42e4f0c263eb603551f977" to package.json

2025-06-26 - 1.1.0 - feat(Smartchok)

Migrate to chokidar 4.x with picomatch glob support, update documentation and tests

- Removed deprecated @tempfix/watcher and added chokidar and picomatch as dependencies in package.json
- Updated Smartchok class to extract base paths and apply custom glob filtering using picomatch
- Revised readme and technical hints to reflect migration to chokidar 4.x and clarify glob pattern support
- Adjusted tests and commit info to align with the updated code structure

2024-05-29 – 1.0.34 – general

This release improves the project description.

- update description

2024-05-06 – 1.0.33 – core

This release includes a mix of bug fixes and configuration updates.

- fix(core): update
- update tsconfig
- update npmextra.json: githost (recorded multiple times)

2024-02-29 – 1.0.32 to 1.0.28 – core fixes

Releases 1.0.32 through 1.0.28 were dedicated to routine core fixes.
(This range covers versions that only included “fix(core): update” changes.)

2024-01-28 – 1.0.27 – core

This release not only fixed core issues but also adjusted the organization scheme.

- fix(core): update
- switch to new org scheme (recorded twice)

2021-12-01 – 1.0.26 to 1.0.14 – core fixes

Releases 1.0.26 through 1.0.14 were devoted to routine core fixes.
(No additional details beyond the core update were recorded.)

2018-02-28 – 1.0.13 to 1.0.11 – ci updates

Releases 1.0.13 through 1.0.11 focused on updating the continuous integration configuration.

- update ci

2017-06-30 – 1.0.10 – general

This release delivered several improvements beyond a simple version bump.

- fix Promise issues
- update test
- update

2017-06-30 – 1.0.9 – general

This release addressed module loading and code hygiene.

- fix loading of rxjs
- clean up code

2017-06-30 – 1.0.8 – general

A targeted update to align output with expectations.

- update to wirj like expected

2017-04-09 – 1.0.7 – ci

An update to the continuous integration configuration.

- update ci

2017-04-09 – 1.0.6 – npm

This release updated extra npm configuration.

- update npmextra.json

2017-02-15 – 1.0.5 – general

Standardization work was undertaken with new organizational practices.

- update to new gitzone standard

2016-11-18 – 1.0.4 – general

This release refreshed dependency settings.

- update dependencies

2016-11-18 – 1.0.3 – general

Readability and developer guidance were improved.

- improve README

2016-11-18 – 1.0.2 – general

Minor documentation and CI configuration enhancements were added.

- add README
- Update .gitlab-ci.yml

2016-09-22 – 1.0.1 – general

A fix was applied to ensure the process exits correctly.

- fix process not exiting problem

2016-09-22 – 1.0.0 – general

The project’s initial setup was established along with CI configuration.

- add gitlab-ci
- initial