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.

Revision #3
Created 2026-03-28 11:10:27 UTC by foss.global Team
Updated 2026-03-28 12:16:42 UTC by foss.global Team