readme.md for @git.zone/tswatch

A powerful, config-driven TypeScript file watcher that automatically recompiles and executes your project when files change. Built for modern TypeScript development with zero-config presets, smart bundling, a built-in dev server with live reload, and deep customization options.

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.

□ Features

- □□ **Config-driven architecture** — Define watchers, bundles, and dev server in `.smartconfig.json`
- ⚡ **Zero-config presets** — Get started instantly with `npm`, `element`, `service`, `website`, and `test` presets
- □□ **Interactive wizard** — Run `tswatch init` to generate configuration interactively
- □□ **Built-in dev server** — Live reload, CORS, compression, SPA fallback out of the box
- □□ **Smart bundling** — TypeScript, HTML, and assets with esbuild/rolldown/rspack integration
- □□ **Debounced execution** — Configurable debounce prevents command spam during rapid file saves
- □□ **Process management** — Automatic restart or queue mode for long-running commands
- □□ **Glob patterns** — Watch any files with flexible pattern matching
- □□ **Graceful shutdown** — Signal-aware process lifecycle with tree-kill for clean teardowns

□□ Installation

```
# Global installation (recommended for CLI usage)
pnpm install -g @git.zone/tswatch

# As a dev dependency
pnpm install --save-dev @git.zone/tswatch
```

□□ Quick Start

Using the Wizard

```
# Run the interactive wizard to create your configuration
tswatch init
```

The wizard guides you through creating a `.smartconfig.json` configuration with your chosen preset or custom watchers.

Using Presets

If you already have a configuration, just run:

```
tswatch
```

This reads your config from `.smartconfig.json` under the `@git.zone/tswatch` key and starts watching.

⚙ Configuration

tswatch uses `.smartconfig.json` for configuration. Add your config under the `@git.zone/tswatch` key:

```
{
  "@git.zone/tswatch": {
    "preset": "npm"
  }
}
```

Available Presets

Preset	Description
`npm`	Watch `ts/` and `test/`, run `npm test` on changes
`test`	Watch `ts/` and `test/`, run `npm run test2` on changes
`service`	Watch `ts/`, restart `npm run startTs` (ideal for backend services)
`element`	Dev server on port 3002 + bundling for web components
`website`	Full-stack: backend restart + frontend bundling + asset processing

Full Configuration Schema

```
{
  "@git.zone/tswatch": {
    "preset": "element",

    "server": {
      "enabled": true,
      "port": 3002,
      "serveDir": "./dist_watch/",
      "liveReload": true,
      "domain": "localhost"
    },

    "bundles": [
      {
        "name": "main-bundle",
        "from": "./ts_web/index.ts",
        "to": "./dist_watch/bundle.js",
        "watchPatterns": ["./ts_web/**/*"],
        "triggerReload": true,
        "bundler": "esbuild",
        "production": false
      },
      {
        "name": "html",
```

```
        "from": "./html/index.html",
        "to": "./dist_watch/index.html",
        "watchPatterns": ["./html/**/*"],
        "triggerReload": true
      }
    ],

    "watchers": [
      {
        "name": "backend-build",
        "watch": "./ts/**/*",
        "command": "npm run build",
        "restart": false,
        "debounce": 300,
        "runOnStart": true
      },
      {
        "name": "tests",
        "watch": ["./ts/**/*", "./test/**/*"],
        "command": "npm test",
        "restart": true,
        "debounce": 300,
        "runOnStart": true
      }
    ]
  }
}
```

Configuration Options

`ITswatchConfig`

Option	Type	Description
`preset`	`string`	Use a preset: `npm`, `test`, `service`, `element`, `website`
`watchers`	`IWatcherConfig[]`	Array of watcher configurations
`server`	`IServerConfig`	Development server configuration
`bundles`	`IBundleConfig[]`	Bundle configurations

> **Tip:** When a preset is specified alongside explicit `watchers`, `bundles`, or `server`, your explicit values take precedence over the preset defaults.

`IWatcherConfig`

Option	Type	Default	Description
`name`	`string`	*required*	Name for logging purposes
`watch`	`string | string[]`	*required*	Glob pattern(s) to watch
`command`	`string`	—	Shell command to execute on changes
`restart`	`boolean`	`true`	Kill previous process before restarting
`debounce`	`number`	`300`	Debounce delay in milliseconds
`runOnStart`	`boolean`	`true`	Run the command immediately on start

`IServerConfig`

Option	Type	Default	Description
`enabled`	`boolean`	*required*	Whether the server is enabled
`port`	`number`	`3002`	Server port
`serveDir`	`string`	`./dist_watch/`	Directory to serve
`liveReload`	`boolean`	`true`	Inject live reload script
`domain`	`string`	`localhost`	Domain name for the dev server

`IBundleConfig`

Option	Type	Default	Description
`name`	`string`	—	Name for logging purposes
`from`	`string`	*required*	Entry point file
`to`	`string`	*required*	Output file
`watchPatterns`	`string[]`	—	Additional patterns to watch
`triggerReload`	`boolean`	`true`	Trigger server reload after bundling

Option	Type	Default	Description
`outputMode`	`'bundle' | 'base64ts'`	`'bundle'`	Output mode for the bundle
`bundler`	`'esbuild' | 'rolldown' | 'rspack'`	`'esbuild'`	Bundler engine to use
`production`	`boolean`	`false`	Enable minification for production builds
`includeFiles`	`(string | { from, to })[]`	—	Additional files to include alongside the bundle
`maxLineLength`	`number`	—	Max chars per line for `base64ts` output mode

□□ CLI Commands

`tswatch`

Runs with configuration from `.smartconfig.json`. If no config exists, launches the interactive wizard automatically.

```
tswatch
```

`tswatch init`

Force-run the configuration wizard (creates or overwrites existing config).

```
tswatch init
```

□□ Programmatic API

Basic Usage with Inline Config

```
import { TsWatch } from '@git.zone/tswatch';

const watcher = new TsWatch({
  watchers: [
```

```
    {
      name: 'my-watcher',
      watch: './src/**/*',
      command: 'npm run build',
      restart: true,
      debounce: 300,
      runOnStart: true,
    },
  ],
});

await watcher.start();

// Later: stop watching
await watcher.stop();
```

Load from Config File

```
import { TsWatch } from '@git.zone/tswatch';

// Load configuration from .smartconfig.json
const watcher = TsWatch.fromConfig();

if (watcher) {
  await watcher.start();
}
```

Using ConfigHandler

```
import { ConfigHandler } from '@git.zone/tswatch';

const configHandler = new ConfigHandler();

// Check if config exists
if (configHandler.hasConfig()) {
  const config = configHandler.loadConfig();
  console.log(config);
```

```
}

// Get available presets
const presets = configHandler.getPresetNames();
// => ['npm', 'test', 'service', 'element', 'website']

// Get a specific preset
const npmPreset = configHandler.getPreset('npm');
```

Using Watcher Directly

For more granular control, use the `Watcher` class:

```
import { Watcher } from '@git.zone/tswatch';

// Create from config object
const watcher = Watcher.fromConfig({
  name: 'my-watcher',
  watch: ['./src/**/*', './lib/**/*'],
  command: 'npm run compile',
  restart: true,
});

await watcher.start();
```

Using Function Callbacks

```
import { Watcher } from '@git.zone/tswatch';

const watcher = new Watcher({
  name: 'custom-handler',
  filePathToWatch: './src/**/*',
  functionToCall: async () => {
    console.log('Files changed! Running custom logic...');
    // Your custom build/test/deploy logic here
  },
  debounce: 500,
```

```
  runOnStart: true,
});

await watcher.start();
```

□□ Project Structure Examples

NPM Package / Node.js Library

```
project/
├── ts/                 # TypeScript source files
├── test/               # Test files
├── package.json        # With "test" script
└── .smartconfig.json   # tswatch config
```

```
{
  "@git.zone/tswatch": {
    "preset": "npm"
  }
}
```

Backend Service

```
project/
├── ts/                 # TypeScript source files
├── package.json        # With "startTs" script
└── .smartconfig.json
```

```
{
  "@git.zone/tswatch": {
    "preset": "service"
  }
}
```

Web Component / Element

```
project/
├── ts/                 # Backend TypeScript (optional)
├── ts_web/             # Frontend TypeScript
├── html/
│   ├── index.ts        # Web entry point
│   └── index.html
├── dist_watch/         # Output (auto-created)
└── .smartconfig.json
```

```
{
  "@git.zone/tswatch": {
    "preset": "element"
  }
}
```

Access your project at `http://localhost:3002`

Full-Stack Website

```
project/
├── ts/                 # Backend TypeScript
├── ts_web/             # Frontend TypeScript
│   └── index.ts
├── html/
│   └── index.html
├── assets/             # Static assets
├── dist_serve/         # Output
└── .smartconfig.json
```

```
{
  "@git.zone/tswatch": {
    "preset": "website"
  }
}
```

□□ Development Server

The built-in development server (powered by `@api.global/typedserver`'s `UtilityWebsiteServer`) is enabled in `element` and `website` presets:

- □□ **Live Reload** — WebSocket-based instant browser refresh on changes (via service worker + devtools injection)
- □□ **No Caching** — Prevents browser caching during development (`Cache-Control: no-store, no-cache` headers)
- □□ **CORS** — Cross-origin requests enabled
- □□ **Compression** — Brotli + gzip compression for faster loading
- □□ **SPA Fallback** — Single-page application routing support
- □□ **Security Headers** — Cross-origin isolation (`COOP`, `COEP`)
- □□ **PWA Manifest** — Auto-generated Progressive Web App manifest
- ⚡ **Service Worker** — Built-in service worker version info for cache busting

Default configuration:

Setting	Default
Port	`3002`
Serve Directory	`./dist_watch/`
Live Reload	Enabled
Domain	`localhost`

□□ Configuration Tips

1. **Use presets for common workflows** — They're battle-tested and cover most use cases
2. **Customize with explicit config** — Override preset defaults by adding explicit `watchers`, `bundles`, or `server` config
3. **Debounce wisely** — Default 300ms works well; increase for slower builds
4. **Use `restart: false`** for one-shot commands (like builds) and `restart: true` for long-running processes (like servers)
5. **Multiple bundlers** — Choose between `esbuild` (fastest), `rolldown` (smallest output), or `rspack` (webpack-compatible) per bundle

License and Legal Information

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 #5
Created 2026-03-28 10:49:33 UTC by foss.global Team
Updated 2026-03-28 12:15:31 UTC by foss.global Team