@git.zone/tswatch

watch typescript projects during development

- readme.md for @git.zone/tswatch
- changelog.md for @git.zone/tswatch

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.

changelog.md for @git.zone/tswatch

2026-03-24 - 3.3.2 - fix(config)

migrate project metadata and documentation to .smartconfig.json

- replace npmextra.json with .smartconfig.json in package files and documentation
- update dependency versions to align with the smartconfig-based setup
- allow watcher executions to be undefined until a process starts

2026-03-24 - 3.3.1 - fix(config)

switch configuration loading and saving from npmextra.json to smartconfig.json

- replace the @push.rocks/npmextra dependency with @push.rocks/smartconfig
- update config handling, CLI messaging, and init flow to use smartconfig.json consistently

2026-03-10 - 3.3.0 - feat(server)

use UtilityWebsiteServer for dev server, add domain option, update docs, and bump dependencies

- Replace TypedServer with UtilityWebsiteServer in TsWatch (start/stop/reload adapted)
- Add server.domain config option and update interfaces
- Update readme to document UtilityWebsiteServer features (PWA, brotli+gzip, service worker, live reload via WebSocket) and add bundle config hints (outputMode, bundler, production)
- Bump dependencies and devDependencies (notable: @api.global/typedserver ^8.4.2, @git.zone/tsbundle ^2.9.1, @push.rocks/* updates)
- Remove unused taskbuffer export from tswatch.plugins.ts

2026-03-03 - 3.2.1 - fix(watcher)

ensure child processes are killed and awaited during shutdown; improve cleanup handlers; bump smartshell dependency to ^3.3.2

- Await child process kill() calls in restart and stop to avoid race conditions and ensure proper termination.
- Add last-resort synchronous SIGKILL in process 'exit' handler to terminate orphaned child processes.
- Make SIGINT and timeout handlers async and await stop() to perform a clean shutdown before exiting.
- Bump @push.rocks/smartshell from ^3.3.0 to ^3.3.2 in package.json.

2026-02-24 - 3.2.0 - feat(bundle)

add configurable bundle output modes and bundler options (support base64ts, production builds, includeFiles, maxLineLength) and route non-default outputs to a CustomBundleHandler

- Added new ITswatchConfig fields: outputMode, bundler, production, includeFiles, maxLineLength
- tswatch now creates a CustomBundleHandler and uses it when outputMode is not 'bundle' (e.g. base64ts)
- Default bundling path now reads bundler and production from bundleConfig (defaults to 'esbuild' and false)
- Bumped dependency @git.zone/tsbundle to ^2.9.0

2026-02-05 - 3.1.0 - feat(dev-server)

add no-cache headers to built-in development server; update docs and bump dependencies

- Introduce noCache: true in ts/tswatch.classes.tswatch.ts to send Cache-Control: no-store, no-cache during development (prevents browser caching).
- Update documentation to describe no-caching behavior (readme.md and readme.hints.md).
- Bump dependencies: @git.zone/tstest ^3.1.8, @types/node ^25.2.1, @push.rocks/npmextra ^5.3.3, @push.rocks/taskbuffer ^4.2.0.

2026-01-24 - 3.0.1 - fix(deps)

downgrade @push.rocks/smartinteract to ^2.0.16

- package.json: @push.rocks/smartinteract ^2.1.0 -> ^2.0.16

2026-01-24 - 3.0.0 - BREAKING CHANGE(tswatch)

refactor tswatch to a config-driven design (load config from npmextra.json) and add interactive init wizard; change TsWatch public API and enhance Watcher behavior

- Switch to config-driven operation: configuration read from npmextra.json under the key @git.zone/tswatch
- Added ConfigHandler for loading/merging presets and new TswatchInit interactive wizard (runInit) to create/save configuration
- Changed TsWatch constructor to accept ITswatchConfig and added TsWatch.fromConfig(cwd?) for loading from npmextra.json
- Significant public API change: previous watchmode string-based constructor/behavior removed/rewired — consumers must migrate to new config-based usage (breaking change)
- Watcher refactor: Watcher.fromConfig, named watchers, array/single path support, debounce, restart/queue handling, runOnStart, safer start/stop behavior and execution tracking
- New TypeScript interfaces: interfaces.config.ts (ITswatchConfig, IWatcherConfig, IServerConfig, IBundleConfig); removed/changed old watchmodes types
- CLI updated to use configuration if present or launch the init wizard; added init command
- Updated tests to cover ConfigHandler, Watcher, and TsWatch config-driven behavior
- Updated dependencies and plugin usage (added @push.rocks/npmextra, @push.rocks/smartinteract; bumped several @git.zone and @push.rocks package versions)

2025-12-11 - 2.3.13 - fix(@push.rocks/smartwatch)

Update @push.rocks/smartwatch dependency to ^6.3.0

- Bump @push.rocks/smartwatch from ^6.2.5 to ^6.3.0 in package.json
- Dependency-only change; no source files modified

2025-12-11 - 2.3.12 - fix(smartwatch)

Bump @push.rocks/smartwatch from ^6.2.4 to ^6.2.5

- Updated dependency @push.rocks/smartwatch to ^6.2.5 in package.json
- No source code changes; dependency version bump only

2025-12-11 - 2.3.11 - fix(typedserver)

Add cross-origin security headers to element mode dev server

- Set crossOriginOpenerPolicy to 'same-origin' and crossOriginEmbedderPolicy to 'require-corp' on the TypedServer used in element mode.
- Improves security and enables cross-origin isolation (e.g. for SharedArrayBuffer) during local development.
- Applies to the development server serving ./dist_watch/ on port 3002.

2025-12-11 - 2.3.10 - fix(dependencies)

Bump dependency versions: @types/node to ^25.0.0 and @push.rocks/smartwatch to ^6.2.4

- Update devDependency @types/node from ^24.10.2 to ^25.0.0
- Update dependency @push.rocks/smartwatch from ^6.2.3 to ^6.2.4

2025-12-11 - 2.3.9 - fix(smartwatch)

Bump @push.rocks/smartwatch dependency to ^6.2.3

- Updated dependency @push.rocks/smartwatch from ^6.2.2 to ^6.2.3
- No source code changes; dependency version bump only

2025-12-11 - 2.3.8 - fix(@push.rocks/smartwatch)

Bump @push.rocks/smartwatch dependency to ^6.2.2

- package.json: updated @push.rocks/smartwatch from ^6.2.1 to ^6.2.2

2025-12-10 - 2.3.7 - fix(smartwatch)

Bump @push.rocks/smartwatch dependency to ^6.2.1

- Updated dependency @push.rocks/smartwatch from ^6.2.0 to ^6.2.1 in package.json

2025-12-10 - 2.3.6 - fix(dependencies)

Bump @types/node to ^24.10.2 and @push.rocks/smartwatch to ^6.2.0

- Dev dependency @types/node updated from ^24.10.1 to ^24.10.2
- Dependency @push.rocks/smartwatch updated from ^6.1.1 to ^6.2.0

2025-12-08 - 2.3.5 - fix(dependencies)

Update @push.rocks/smartwatch dependency to ^6.1.1

- Bump @push.rocks/smartwatch from ^6.1.0 to ^6.1.1 in package.json
- Only package.json changed; no source code modifications

2025-12-08 - 2.3.4 - fix(dependencies.@push.rocks/smartwatch)

Bump @push.rocks/smartwatch dependency to ^6.1.0

- Updated package.json dependency @push.rocks/smartwatch from ^6.0.0 to ^6.1.0

2025-12-08 - 2.3.3 - fix(dependencies)

Bump dependencies: @api.global/typedserver to ^7.11.1 and @push.rocks/smartwatch to ^6.0.0

- Updated @api.global/typedserver from ^7.11.0 to ^7.11.1 (patch).
- Updated @push.rocks/smartwatch from ^5.1.0 to ^6.0.0 (major). Verify compatibility as this may include breaking changes in that dependency.
- Change is limited to package.json (dependency version updates).

2025-12-08 - 2.3.2 - fix(smartwatch)

Bump @push.rocks/smartwatch dependency to ^5.1.0

- Updated dependency @push.rocks/smartwatch from ^5.0.0 to ^5.1.0 in package.json

2025-12-08 - 2.3.1 - fix(element)

Enable SPA fallback in element dev server

- Add spaFallback: true to the TypedServer configuration used in element mode.
- Improves developer experience for single-page apps by serving the index file for unknown routes during development and supporting client-side routing.

2025-12-08 - 2.3.0 - feat(typedserver)

Enable compression for element development server and update @api.global/typedserver dependency

- Enable HTTP compression (compression: true) for the element mode development server (TypedServer) to improve asset delivery during development.
- Bump dependency @api.global/typedserver from ^7.10.2 to ^7.11.0 in package.json.

2025-12-08 - 2.2.5 - fix(typedserver)

Update @api.global/typedserver to ^7.10.2 and remove deprecated compression options from TypedServer initialization

- Bump @api.global/typedserver dependency from ^7.4.1 to ^7.10.2.
- Remove enableCompression and preferredCompressionMethod options when creating TypedServer in element mode to be compatible with the newer API.

2025-12-04 - 2.2.4 - fix(dependencies)

Bump dependency versions: @api.global/typedserver, @git.zone/tsbundle, @push.rocks/smartfs, @push.rocks/taskbuffer

- Upgrade @api.global/typedserver from ^3.0.80 to ^7.4.1
- Upgrade @git.zone/tsbundle from ^2.6.2 to ^2.6.3
- Upgrade @push.rocks/smartfs from ^1.1.3 to ^1.2.0
- Upgrade @push.rocks/taskbuffer from ^3.4.0 to ^3.5.0

2025-12-04 - 2.2.3 - fix(tswatch.classes.watcher)

Convert directory watch paths to glob patterns for smartwatch compatibility

- Watcher: convert directory paths to recursive glob patterns before adding them to Smartwatch so directories are watched recursively (e.g. /path/to/dir/ -> /path/to/dir/**/*).
- Readme: added migration note explaining that directory paths are converted to glob patterns for smartwatch compatibility.

2025-12-01 - 2.2.2 - fix(core)

Replace smartchok/smartfile with smartwatch/smartfs, update watcher and plugins, and bump dependencies

- Replaced @push.rocks/smartchok with @push.rocks/smartwatch and updated Watcher to use Smartwatch (add/start/stop/getObservableFor).
- Replaced @push.rocks/smartfile with @push.rocks/smartfs and added SmartFs usage (SmartFsProviderNode) plus a listFolders helper used by TsWatch.
- Updated tswatch.plugins.ts to export smartfs and smartwatch and removed smartchok/smartfile exports.
- Updated tswatch.classes.watcher.ts and tswatch.classes.tswatch.ts to use the new smartwatch/smartfs APIs and adjusted directory listing and watcher logic accordingly.
- Bumped several devDependencies and dependencies in package.json (tsbuild, tstest, @api.global/typedserver, @git.zone/tsbundle, @git.zone/tsrun, @push.rocks/* packages).

- Documentation updates: readme.md and readme.hints.md include migration notes and Issue Reporting & Security section.

2025-07-29 - 2.2.1 - fix(exports)

Fix package.json exports field syntax

- Fixed exports field syntax from "./" to "." for proper module resolution

2025-07-29 - 2.2.0 - feat(exports)

Modernize package exports and expose watcher class

- Updated package.json to use modern exports field instead of main/typings
- Exposed TsWatch watcher class through index exports

2025-07-29 - 2.1.3 - fix(documentation)

Update and align documentation with internal CLI and project structure

- Refined readme.md and readme.hints.md to clearly describe the various watch modes and project setup
- Ensured the CLI command mappings in tswatch.cli.ts are documented for consistent usage
- Included update to internal commit information file for clarity

2025-06-26 - 2.1.2 - fix(dependencies)

Update @push.rocks/smartchok dependency to ^1.1.1

- Bump @push.rocks/smartchok version from ^1.0.34 to ^1.1.1 in package.json

2025-06-26 - 2.1.1 - fix(deps)

Update dependency versions and test import paths for enhanced stability

- Bump @git.zone/tsbuild from 2.2.1 to 2.6.4
- Upgrade @git.zone/tstest from 1.0.96 to 2.3.1 and update test import path
- Update @api.global/typedserver from 3.0.55 to 3.0.74
- Update @git.zone/tsbundle from 2.2.1 to 2.5.1
- Bump @push.rocks/lik from 6.1.0 to 6.2.2
- Update @push.rocks/smartfile from 11.1.6 to 11.2.5
- Upgrade @push.rocks/smartlog from 3.0.7 to 3.1.8
- Bump @push.rocks/smartshell from 3.2.2 to 3.2.3
- Upgrade @types/node from 22.12.0 to 24.0.4
- Add packageManager field in package.json for pnpm v10.11.0

2025-01-29 - 2.1.0 - feat(CI)

Add Continuous Integration workflows for Gitea with Docker-based setup

- Added new CI workflows for handling both regular and tagged pushes in Gitea.
- Integrated security audits and setup tasks using Docker images in the CI workflows.
- Ensured that CI includes testing, building, and releasing steps as per tag events.

2025-01-29 - 2.0.39 - fix(package.json)

Add pnpm overrides configuration for peek-readable package

- Added pnpm overrides section in package.json
- Specified version 5.3.1 for peek-readable package

2025-01-29 - 2.0.38 - fix(core)

Updated dependencies and added assetsHandler instantiation

- Updated various dependencies in package.json to latest versions.
- Added assetsHandler instantiation in TsWatch class to improve functionality.

2024-12-09 - 2.0.37 - fix(core)

Refactor TsWatch class to improve website execution handling

- Removed unnecessary Smartshell instance creation in TsWatch class.
- Ensured websiteExecution restarts and website bundle reloads in watcher function.

2024-12-09 - 2.0.36 - fix(dependencies)

Update @push.rocks/smartshell dependency version

- Upgrade @push.rocks/smartshell to version ^3.2.0 from ^3.1.0 in package.json

2024-12-09 - 2.0.35 - fix(core)

Fixed website watch mode execution method

- Replaced direct shell command execution with SmartExecution instance for website mode.
- Updated dependency version for @push.rocks/smartshell to ^3.1.0.

2024-12-04 - 2.0.34 - fix(TsWatch)

Fix reloading issue for tsfolder changes in element mode.

- Adjusted the function call from 'this.typedserver.reload()' to 'bundleAndReloadElement()' to ensure proper bundle handling in 'element' mode.

2024-12-04 - 2.0.33 - fix(core)

Improve async handling in TsWatch class for element and website modes

- Ensured proper asynchronous execution for 'element' and 'website' watch modes.
- Replaced console log with logger for consistency.

2024-12-04 - 2.0.32 - fix(core)

Minor improvements and dependency updates

- Updated dependencies for improved performance
- Refined TypeScript project watching logic

2024-12-04 - 2.0.31 - fix(Watcher)

Add missing logger message in Watcher class for start method

- Added a log message when starting a watcher to track file path being watched

2024-12-04 - 2.0.30 - fix(cli)

Fix incorrect watch mode and update npm test command.

- Updated the npm test command in .vscode/launch.json for streamlined launching.
- Corrected the watch mode from 'echoSomething' to 'echo' in test/test.ts.

2024-12-04 - 2.0.29 - fix(core)

No changes detected in the codebase.

2024-12-04 - 2.0.28 - fix(tswatch)

Add logging to notify folder watcher creation and building processes

- Logging added to notify the creation of folder watchers.

- Logging added to indicate the start of the building process for folders.

2024-12-04 - 2.0.27 - fix(core)

Refactor watch mode commands and exports

- Updated watch mode names in CLI and type definitions for consistency.
- Added export for smartfile plugin.
- Ensured updated command logic is applied during CLI operations.

2024-12-04 - 2.0.26 - fix(core)

Refactor watch modes and update dependencies

- Updated dependencies in package.json
- Refactored watch mode names in interfaces and classes
- Refactored CLI commands to use new watch mode names
- Added import for smartfile in tswatch.plugins.ts

2024-10-27 - 2.0.25 - fix(typescript)

Remove unnecessary reference types in TypeScript declaration files

- Removed unnecessary comment line from TypeScript declaration files.

2024-10-27 - 2.0.24 - fix(core)

Remove .gitlab-ci.yml and update dependencies in package.json

- Deleted .gitlab-ci.yml which contained CI/CD configuration settings.
- Updated various dependencies and devDependencies in package.json to newer versions.

2024-01-28 to 2024-01-09 - 2.0.14 to 2.0.23 - Core

Several updates to the core functionality.

- Updated core functionality for efficiency and stability.

2023-09-21 to 2023-01-28 - 2.0.8 to 2.0.13 - Core

Core system updates and fixes.

- Continuous updates to core functionality for improved performance.

2023-03-31 to 2022-08-04 - 2.0.0 to 2.0.7 - Core

Enhanced core operations.

- Regular core updates to address performance and reliability.

2022-05-04 - 1.0.81 - Core

Introduction of a breaking change in the core system.

- Switched to gitzone v2 generation tools.

2022-04-21 to 2022-03-18 - 1.0.62 to 1.0.80 - Core

Routine updates to core operations.

- Various core fixes enhancing stability and function.

2022-03-14 to 2022-03-18 - 1.0.61 to 1.0.75 - Core

Further adjustments to core functionality.

- Fixes across multiple modules in core.

2021-08-17 to 2020-07-07 - 1.0.46 to 1.0.60 - Core

Core component updates.

- Consistent core updates to refine performance.

2020-07-04 to 2020-05-22 - 1.0.39 to 1.0.45 - Core

Incremental updates to core systems.

- Enhanced features within the core for better integration.

2020-03-13 to 2020-03-05 - 1.0.30 to 1.0.38 - Core

Stabilizing and refining core functions.

- Series of core fixes improving overall system durability.

2019-10-14 to 2019-10-12 - 1.0.21 to 1.0.29 - Core

Tweaks and enhancements to the core mechanics.

- Continued improvements in core system support services.

2019-05-28 to 2019-05-06 - 1.0.10 to 1.0.20 - Core

Core module enhancements and fixes.

- Several iterations to the core functionality for increased resilience.

2019-05-06 to 2019-05-08 - 1.0.5 to 1.0.9 - Core

Regular updates ensuring core integrity.

- Stability improvements within the core services.

2018-10-28 - 1.0.1 to 1.0.3 - Core

Initial setup and fixes to the core.

- Foundational updates to core functionality.