@git.zone/tsbundle

a bundler based on esbuild for easy project bundling

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

readme.md for @git.zone/tsbundle

A powerful multi-bundler tool supporting **esbuild**, **rolldown**, and **rspack** for painless bundling of web projects.

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.

Installation

```
# Global installation for CLI usage
pnpm add -g @git.zone/tsbundle

# Local installation for project usage
pnpm add --save-dev @git.zone/tsbundle
```

Quick Start

Interactive Setup

The easiest way to get started is with the interactive wizard:

```
tsbundle init
```

This guides you through setting up your bundle configuration with preset options:

- **element** - Web component / element bundle (`./ts_web/index.ts` -> `./dist_bundle/bundle.js`)
- **website** - Full website with HTML and assets (`./ts_web/index.ts` -> `./dist_serve/bundle.js`)
- **npm** - NPM package bundle (`./ts/index.ts` -> `./dist_bundle/bundle.js`)
- **custom** - Configure everything manually

Build Your Bundles

Once configured, simply run:

```
tsbundle
```

Your bundles will be built according to your `smartconfig.json` configuration.

CLI Commands

Command	Description
`tsbundle`	Build all bundles from `smartconfig.json` configuration
`tsbundle custom`	Same as above (explicit)
`tsbundle init`	Interactive wizard to create/update bundle configuration

Configuration

tsbundle uses `smartconfig.json` for configuration. Here's an example:

```
{
  "@git.zone/tsbundle": {
    "bundles": [
      {
        "from": "./ts_web/index.ts",
        "to": "./dist_bundle/bundle.js",
        "outputMode": "bundle",
        "bundler": "esbuild",
        "production": false
      },
```

```
      {
        "from": "./ts_web/index.ts",
        "to": "./dist_serve/bundle.js",
        "outputMode": "bundle",
        "bundler": "esbuild",
        "includeFiles": ["./html/**/*.html", "./assets/**/*"]
      }
    ]
  }
}
```

Bundle Configuration Options

Option	Type	Default	Description
`from`	`string`	-	Entry point TypeScript file
`to`	`string`	-	Output file path
`outputMode`	`"bundle"` | `"base64ts"`	`"bundle"`	Output format (see below)
`bundler`	`"esbuild"` | `"rolldown"` | `"rspack"`	`"esbuild"`	Which bundler to use
`production`	`boolean`	`false`	Enable minification
`includeFiles`	`string[]`	`[]`	Glob patterns for additional files (HTML, assets)
`maxLineLength`	`number`	`0` (unlimited)	For `base64ts` mode: max chars per line in output

Output Modes

`bundle` (default)

Standard JavaScript bundle output. Additional files specified in `includeFiles` are copied to the output directory.

`base64ts`

Generates a TypeScript file with base64-encoded content - perfect for **Deno compile** scenarios where you need everything embedded in a single executable:

```
// Auto-generated by tsbundle
export const files: { path: string; contentBase64: string }[] = [
  { path: "bundle.js", contentBase64: "Y29uc3QgaGVsbG8gPSAid29ybGQi..." },
  { path: "index.html", contentBase64: "PCFET0NUWVBFIGh0bWw+..." },
];
```

If you're working with AI tools that have line length limitations, set `maxLineLength` (e.g., `200`) to split long base64 strings across multiple lines.

Available Bundlers

tsbundle supports three modern bundlers, each with different strengths:

Bundler	Speed	Bundle Size	Best For
esbuild	Fastest	Medium	Development, quick iterations
rolldown	Fast	Smallest	Production builds, tree-shaking
rspack	Fast	Largest (webpack runtime)	Webpack compatibility

API Usage

TsBundle Class

The core bundling class, usable programmatically:

```
import { TsBundle } from '@git.zone/tsbundle';

const bundler = new TsBundle();

await bundler.build(
  process.cwd(),           // Working directory
  './src/index.ts',        // Entry point
  './dist/bundle.js',      // Output path
  {
    bundler: 'esbuild',    // 'esbuild' | 'rolldown' | 'rspack'
```

```
    production: true
  }
);
```

Each bundler runs in a separate child process via `smartspawn.ThreadSimple`, keeping the main process clean and isolated from bundler-specific dependencies.

HtmlHandler Class

Process and optionally minify HTML files:

```
import { HtmlHandler } from '@git.zone/tsbundle';

const htmlHandler = new HtmlHandler();

await htmlHandler.processHtml({
  from: './html/index.html',
  to: './dist/index.html',
  minify: true
});
```

AssetsHandler Class

Copy static assets between directories:

```
import { AssetsHandler } from '@git.zone/tsbundle';

const assetsHandler = new AssetsHandler();

await assetsHandler.processAssets({
  from: './assets',
  to: './dist/assets'
});
```

Base64TsOutput Class

Generate TypeScript files with base64-encoded content for embedding:

```
import { Base64TsOutput } from '@git.zone/tsbundle';

const output = new Base64TsOutput(process.cwd());
output.addFile('bundle.js', bundleBuffer);
await output.addFilesFromGlob('./html/**/*.html');
await output.writeToFile('./ts/embedded-bundle.ts', 200); // optional maxLineLength
```

CustomBundleHandler Class

Process multiple bundle configurations from `smartconfig.json`:

```
import { CustomBundleHandler } from '@git.zone/tsbundle';

const handler = new CustomBundleHandler(process.cwd());
const hasConfig = await handler.loadConfig();
if (hasConfig) {
  await handler.processAllBundles();
}
```

Embedding for Deno Compile

For single-executable scenarios with Deno:

```
tsbundle init
# Select "custom", set outputMode to "base64ts"
```

Config:

```
{
  "@git.zone/tsbundle": {
    "bundles": [
      {
        "from": "./ts_web/index.ts",
        "to": "./ts/embedded-bundle.ts",
        "outputMode": "base64ts",
        "bundler": "esbuild",
        "production": true,
        "includeFiles": ["./html/index.html"],
```

```
        "maxLineLength": 200
      }
    ]
  }
}
```

Then in your Deno app:

```
import { files } from './ts/embedded-bundle.ts';

// Decode and serve your embedded files
const bundle = files.find(f => f.path === 'bundle.js');
const html = files.find(f => f.path === 'html/index.html');

const bundleContent = atob(bundle.contentBase64);
const htmlContent = atob(html.contentBase64);
```

Project Structure Recommendations

```
your-project/
├── ts_web/              # Web bundle entry points
│   └── index.ts
├── ts/                  # Library/node entry points
│   └── index.ts
├── html/                # HTML templates
│   └── index.html
├── assets/              # Static assets (images, fonts, etc.)
├── dist_bundle/         # Output for element/npm bundles
├── dist_serve/          # Output for website bundles
└── smartconfig.json     # tsbundle configuration
```

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 @git.zone/tsbundle

2026-03-24 - 2.10.0 - feat(config)

migrate project configuration to smartconfig.json and update bundler dependencies

- replace npmextra.json with .smartconfig.json and update CLI, init, and custom bundle loading references
- rename the smartconfig plugin export and load tsbundle configuration through the new config file
- update rolldown output settings to use codeSplitting: false instead of the deprecated inlineDynamicImports option
- refresh core build and bundler dependencies including rolldown, rspack, esbuild, TypeScript, and smartfs
- revise README documentation to reflect smartconfig.json usage and current bundler capabilities

2026-03-24 - 2.9.3 - fix(config)

migrate configuration loading and init output from npmextra.json to smartconfig.json

- replace @push.rocks/npmextra with @push.rocks/smartconfig
- load project configuration via Smartconfig in the custom module
- update init wizard to read from and write to smartconfig.json and adjust user-facing messages accordingly

2026-03-11 - 2.9.2 - fix(mod_esbuild)

preserve function and class names in esbuild output by enabling keepNames in both dev and prod configs

- Added keepNames: true to esbuild options in ts/mod_esbuild/index.child.ts for the non-minified/dev build
- Added keepNames: true to esbuild options in ts/mod_esbuild/index.child.ts for the minified/production build to improve stack traces, debugging, and runtime reflection

2026-03-05 - 2.9.1 - fix(mod_custom)

use absolute smartfs entry.path instead of joining with dirPath when building fullPath

- entry.path is already absolute (from smartfs); avoid joining it with dirPath which produced incorrect paths
- Fixes file copy path construction so plugins.fs.file(fullPath).copy(destPath) uses the correct source path

2026-02-24 - 2.9.0 - feat(exports)

expose mod_custom, mod_output and interfaces from entry; make processSingleBundle public

- Exported ./mod_custom, ./mod_output and ./interfaces from ts/index.ts to expose these modules in the public API.
- Changed processSingleBundle in ts/mod_custom/index.ts from private to public to allow programmatic invocation.
- Non-breaking API expansion; recommend a minor version bump.

2026-02-24 - 2.8.4 - fix()

no changes — empty diff, nothing to commit

- Diff contained no modifications; no release required

2026-01-23 - 2.8.3 - fix(mod_output)

use pattern base dir when computing relative paths for files to serve

- Compute relativePath using the pattern base directory (dirPath) instead of this.cwd to ensure correct web-serving paths for absolute or relative entry.path values.
- File changed: ts/mod_output/index.ts — replaces plugins.path.relative(this.cwd, fullPath) with plugins.path.relative(dirPath, fullPath) and adds clarifying comment.

2026-01-23 - 2.8.2 - fix(mod_output)

resolve absolute and relative entry.path correctly when adding files

- Add check for plugins.path.isAbsolute(entry.path) to avoid incorrectly joining absolute paths with dirPath
- Use entry.path directly when it's absolute, otherwise join with dirPath
- Ensures correct relativePath calculation and prevents invalid file reads

2026-01-12 - 2.8.1 - fix(readme)

document maxLineLength option for base64ts output and add example and tip

- Add documented `maxLineLength` configuration option (number, default 0 = unlimited) for `base64ts` output.
- Include example config showing `maxLineLength: 200`.
- Add a tip recommending setting `maxLineLength` to split long base64 strings when using AI tools with line-length limits.

2026-01-12 - 2.8.0 - feat(tsbundle)

add configurable maxLineLength for base64ts output and improve build/error handling in child builds

- Add optional maxLineLength?: number to IBundleConfig to control max characters per line for base64ts output (0 or undefined = unlimited).
- Support splitting base64 strings when maxLineLength is specified; generateTypeScript(maxLineLength?) and writeToFile(outputPath, maxLineLength?) updated to accept and apply this setting.
- Pass bundleConfig.maxLineLength through in mod_custom so base64ts output respects bundle configuration.
- Wrap TsBundle.build in mod_custom with try/catch to log failures and skip output handling when build fails.
- tsbundle.class now rejects the bundle promise when the child process exits with a non-zero status.
- mod_esbuild child process now awaits builds, exits with appropriate success/failure codes, and formats esbuild errors for clearer console output.

2026-01-12 - 2.7.4 - fix(deps)

bump @push.rocks/smartcli dependency to ^4.0.20

- @push.rocks/smartcli: ^4.0.19 → ^4.0.20
- Patch-level dependency update with no breaking changes

2026-01-12 - 2.7.3 - fix(mod_output)

wrap long base64 file contents and format generated TypeScript output to avoid extremely long lines

- Introduce MAX_LINE_LENGTH (200) and formatBase64 to split long base64 strings into chunks and join them with string concatenation
- Emit the files array as nicely indented object entries instead of a single JSON.stringify output to improve readability and avoid extremely long lines

2026-01-12 - 2.7.2 - fix(readme)

update README to add interactive setup (tsbundle init), expand quick start and usage examples, include pnpm install, document embedding/base64ts output with Deno example, add project structure recommendations, and clarify license/trademark wording

- Add interactive setup command: `tsbundle init` and updated Quick Start flow
- Include pnpm global install instruction in installation section
- Replace API example with clarified build instructions and new bundle presets (element, website, npm, custom)
- Document embedded bundle output (base64ts) and Deno usage example for decoding/serving files
- Add recommended project structure and formatting improvements (emoji/icons, section reorganizations)
- Clarify license link and expand trademark wording to mention third-party trademarks and approval requirements

2026-01-11 - 2.7.1 - fix(package.json)

update repository URL to code.foss.global

- repository.url changed from https://gitlab.com/gitzone/tsbundle.git to https://code.foss.global/git.zone/tsbundle.git
- bugs.url in package.json still points to https://gitlab.com/gitzone/tsbundle/issues

2026-01-11 - 2.7.0 - feat(tsbundle)

add npmextra-driven custom bundles, base64-ts output and interactive init wizard

- Add CustomBundleHandler to process bundle configs from npmextra.json (ts/mod_custom/*)
- Implement Base64TsOutput for embedding bundled files as base64 TypeScript (ts/mod_output/*)
- Add interactive 'init' wizard to scaffold npmextra.json bundle presets (ts/mod_init/*)
- Wire new features into CLI: default command runs custom bundles, added 'custom' and 'init' commands (ts/tsbundle.cli.ts)
- Expose @push.rocks/npmextra and @push.rocks/smartinteract in plugins and add them to package.json dependencies
- Update npmextra.json structure and release registries configuration

2025-12-02 - 2.6.3 - fix(cli)

Use basename when collecting HTML files for the website CLI command to ensure correct relative paths

- ts/tsbundle.cli.ts: use plugins.path.basename(entry.path) when building htmlFiles list instead of full entry.path
- Prevents incorrect paths when calling HtmlHandler.processHtml with './html/' and ensures HTML files are processed from the expected relative html directory

2025-11-30 - 2.6.2 - fix(deps)

Bump dependencies and migrate test fixtures to ts_web

- Bumped devDependencies: @git.zone/tsbuild ^3.1.0 -> ^3.1.2, @types/node ^22.12.0 -> ^24.10.1
- Bumped runtime dependencies: @push.rocks/smartfs ^1.1.0 -> ^1.1.3, @rspack/core ^1.6.4 -> ^1.6.5, rolldown 1.0.0-beta.51 -> 1.0.0-beta.52
- Reworked tests: removed test/test-decorators.ts and test/ts_web/test-lit.ts; added test/ts_web/fixture-decorators.ts and test/ts_web/fixture-lit.ts (moved fixtures into ts_web)
- Updated package.json to include the dependency version bumps

2025-11-23 - 2.6.1 - fix(license)

Update copyright holder in license to Task Venture Capital GmbH

- Replaced the copyright owner in the license file from Lossless GmbH to Task Venture Capital GmbH

2025-11-23 - 2.6.0 - feat(core)

Integrate Rolldown as optional bundler, migrate filesystem to smartfs, and update bundler/tooling

- Add optional 'rolldown' bundler and wiring in TsBundle (supports --bundler=rolldown)
- Migrate filesystem usage from @push.rocks/smartfile to @push.rocks/smartfs and provide a shared async SmartFs instance
- Refactor HtmlHandler and AssetsHandler to use async smartfs APIs and improve HTML/asset processing (create dirs, write files, delete/replace targets)
- Update bundler child processes to read tsconfig async for alias resolution and normalize argument handling

- Bump dev and runtime dependencies: tsbuild, tsrun, tstest, rolldown, rspack, esbuild, typescript and related packages
- Add repository/metadata fields and pnpm section to package.json
- Add CI workflow definitions (.gitea/workflows) and docs build script (buildDocs)
- Update TypeScript config target to ES2022, adjust module settings and baseUrl/paths
- Small test and formatting fixes (test runners, HTML test, decorator test output formatting)

2025-11-17 - 2.5.2 - fix(tsconfig)

Update TypeScript configs to ES2022 and remove deprecated compiler flags

- assets/tsconfig.json: set target and module to ES2022 (was ES2020)
- assets/tsconfig.json and tsconfig.json: remove experimentalDecorators and useDefineForClassFields flags to align with updated TS setup

2025-06-26 - 2.5.1 - fix(readme)

Update license and legal information section in readme

- Replaced contribution guidelines with detailed legal and trademark information
- Clarified MIT license usage and limitations regarding trade names and trademarks
- Added company information for Task Venture Capital GmbH

2025-06-26 - 2.5.0 - feat(documentation)

Improve README with comprehensive installation, usage, and API reference details

- Updated installation instructions for both global and local setups
- Added a quick start guide featuring CLI usage and API examples
- Enhanced sections for available bundlers and specialized CLI commands
- Expanded API reference with detailed examples for TsBundle, HtmlHandler, and AssetsHandler
- Reorganized content to improve clarity and best practices guidance

2025-06-26 - 2.4.1 - fix(tests)

Improve decorator tests and add LitElement component tests for better validation

- Refactored test-decorators.ts to robustly verify that the sealed decorator prevents prototype modifications
- Added test-lit.ts to ensure LitElement component renders correctly and increments counter on click

2025-06-19 - 2.4.0 - feat(bundler)

Introduce rspack bundler support and update multi-bundler workflow

- Added full support for rspack with its own implementation in ts/mod_rspack
- Updated package.json: new dependency on @rspack/core and revised description
- Refactored bundler types and switch statement to remove deprecated rollup and parcel options
- Modified test suite to include tests for esbuild, rolldown, and rspack with bundle size comparisons
- Adjusted output configuration for esbuild and rolldown for dynamic naming and inline dynamic imports

2025-06-19 - 2.3.0 - feat(bundler)

Integrate rolldown bundler support and update bundler selection logic

- Added rolldown dependency to package.json
- Extended ICliOptions to include 'rolldown' as a valid bundler option
- Created ts/mod_rolldown module with buildTest and buildProduction implementations
- Updated getBundlerPath in tsbundle.class.tsbundle.ts to route to new rolldown module
- Revised readme and hints documentation for rolldown usage

2025-01-29 - 2.2.5 - fix(mod_assets)

Fix async handling in asset processing

- Ensured that the empty directory operation is awaited in the asset processing workflow.

2025-01-29 - 2.2.4 - fix(mod_assets)

Fix logging message in ensureAssetsDir to correctly state when directory is created

- Corrected logging output in ensureAssetsDir method to indicate directory creation.

2025-01-29 - 2.2.3 - fix(mod_assets)

Fix issue with asset directory copy

- Updated dependency '@push.rocks/smartfile' to version '^11.2.0'
- Ensure target directory is properly replaced when copying assets

2025-01-29 - 2.2.2 - fix(dependencies)

Update smartfile dependency and fix spacing issue in assets module

- Updated @push.rocks/smartfile from ^11.1.6 to ^11.1.8
- Fixed a spacing issue in the processAssets function within the assets module

2025-01-29 - 2.2.1 - fix(index)

Export mod_assets for programmatic use

- Added export for mod_assets/index in ts/index.ts to make it usable programmatically.

2025-01-29 - 2.2.0 - feat(AssetsHandler)

Add asset handling to the CLI workflow

- Introduced AssetsHandler class for managing asset directories and files.
- Updated tsbundle.cli.ts to include asset processing in the 'website' command.

2025-01-28 - 2.1.1 - fix(core)

Update dependencies and remove GitLab CI configuration.

- Updated several devDependencies to newer versions for improved stability and performance.
- Updated core dependencies including esbuild and TypeScript.
- Removed the .gitlab-ci.yml file, which could suggest a change in continuous integration setup.

2024-10-27 - 2.1.0 - feat(mod_esbuild)

Add alias support to esbuild bundling process

- Updated dependencies in package.json to latest versions.
- Improved build process by adding alias resolution based on tsconfig.json settings in esbuild.

2022-05-04 - 2.0.0-2.0.1 - Breaking and Fix Changes

Released version 2.0.0 with breaking changes and subsequent fixes.

- BREAKING CHANGE(core): Removed parcel and rollup
- fix(core): Addressed initial issues in new major version

2023-10-03 - 2.0.10 - Fix Updates

Ongoing updates and improvements.

- fix(core): General updates and enhancements

2024-01-10 - 2.0.11-2.0.15 - Minor Fixes

Cumulative fixes and updates from recent releases.

- fix(core): Continuous improvement cycle across versions