readme.md for @push.rocks/smartgulp

lightweight gulp replacement

Install

To install `@push.rocks/smartgulp`, use the following command in your terminal:

```
npm install @push.rocks/smartgulp --save-dev
```

This will add `smartgulp` to your project's `devDependencies`. `smartgulp` is designed to be a lightweight replacement for `gulp`, thus assuming its role in development processes rather than in production environments.

Usage

In this section, we'll dive into how to effectively utilize `@push.rocks/smartgulp` within your project. `smartgulp` aims to present a simplified interface for handling file streams and processing, similar to traditional `gulp` but with a modernized API and reduced complexity.

Basic Stream Creation and File Manipulation

First, let's demonstrate how to create a stream from source files and then manipulate these files, ultimately saving the transformed files to a desired directory.

```
// Import smartgulp methods
import { src, dest } from '@push.rocks/smartgulp';

// Create a stream from source files using the `src` function
```

```
src(['src/**/*.ts']) // Specify the glob pattern
  .pipe(
    // Example transformation: Convert TypeScript to JavaScript
    // Use your desired transformer here
    ts() // Assuming `ts()` is a function returning a Transform stream
  )
  .pipe(
    // Save the transformed files to the 'dist' directory
    dest('dist')
  );
```

In the example above, `src(['src/**/*.ts'])` creates a stream that reads all TypeScript files in the `src` directory. Each file is then transformed (for instance, compiled from TypeScript to JavaScript, though the actual transformation function `ts()` is illustrative) and finally written to the `dist` directory via `dest('dist')`.

Advanced Usage: Custom Transformations

`smartgulp` can be extended with custom transformations, allowing you to perform virtually any file manipulation task. Here's how you can create and use custom transform streams:

```
import { src, dest } from '@push.rocks/smartgulp';
import { Transform } from 'stream';

// Creating a custom transform to prefix file contents
const prefixTransform = new Transform({
  objectMode: true,
  transform(file, encoding, callback) {
    const prefix = '/* Prefixed content */\n';
    file.contents = Buffer.from(prefix + file.contents.toString());
    callback(null, file); // Pass the modified file along the stream
  }
});

// Usage in a pipeline
src(['src/**/*.js']) // Adjust glob pattern as needed
  .pipe(prefixTransform) // Apply custom prefixing
  .pipe(dest('dist')); // Output to 'dist' directory
```

In this scenario, `prefixTransform` is a custom transformation that prepends a comment to the contents of each `.js` file fetched through `src(['src/**/*.js'])`. After transformation, the files are written to the `dist` directory.

Combining with Other Libraries

`@push.rocks/smartgulp` can be seamlessly integrated with other libraries, making it easy to enhance your build pipeline with additional functionality like linting, minification, and more.

```
import { src, dest } from '@push.rocks/smartgulp';
import someOtherLib from 'some-other-lib';

src(['src/**/*.css']) // Example with CSS files
  .pipe(
    // Assuming `someOtherLib` provides a method for CSS minification
    someOtherLib.minifyCSS()
  )
  .pipe(dest('dist'));
```

Conclusion

`@push.rocks/smartgulp` offers a simplified, promise-based approach to stream processing and file manipulation. By leveraging its functionality along with custom and third-party transforms, you can create powerful and efficient build pipelines tailored to your project's needs. Whether you're compiling, bundling, minifying, or simply copying files, `smartgulp` provides a lightweight, flexible foundation for your automated workflows.

License and Legal Information

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 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, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require 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 11:10:41 UTC by foss.global Team
Updated 2026-03-28 12:17:29 UTC by foss.global Team