@push.rocks/smartspawn

A node module for smart subprocess handling with support for promises and streamlined subprocess communication.

- readme.md for @push.rocks/smartspawn

readme.md for @push.rocks/smartspawn

Given the provided files and their contents, the comprehensive documentation for using `@push.rocks/smartspawn` with TypeScript and in a manner that provides thorough examples and descriptions of features and scenarios is extensive. However, I'll outline a concise and informative usage guide that can serve as a helpful starting point. For an in-depth understanding and a complete reference to all features, further reading, exploration, and experimentation with the library are recommended.

@push.rocks/smartspawn

smart subprocess handling

Install

To install `@push.rocks/smartspawn`, open your terminal and run the following command:

```
npm install @push.rocks/smartspawn
```

This command fetches the package from npm and adds it to your project's dependencies.

Usage

This guide uses TypeScript and ESM syntax to demonstrate how to utilize `@push.rocks/smartspawn` for smart subprocess handling in your application.

First, ensure that your TypeScript environment is set up to support ECMAScript modules (ESM). Your `tsconfig.json` should include:

```
{
  "compilerOptions": {
```

```
    "module": "ESNext",
    "target": "ESNext",
    "moduleResolution": "node"
  }
}
```

Starting and Stopping a Simple Thread

`ThreadSimple` class allows you to manage subprocesses effectively. Here's how to start and stop a simple thread:

```
// Import the necessary components from the library
import { ThreadSimple } from '@push.rocks/smartspawn';

// Define a function that creates, starts, and stops a thread
async function demonstrateThreadSimple() {
    // Create an instance of ThreadSimple
    const mySimpleThread = new ThreadSimple('./path/to/worker.js');

    // Start the thread
    await mySimpleThread.start();
    console.log('Thread has started.');

    // Stop the thread
    await mySimpleThread.stop();
    console.log('Thread has stopped.');
}

// Call the function to demonstrate its functionality
demonstrateThreadSimple();
```

The `./path/to/worker.js` should be the path to your worker script that you want to execute in a separate process.

Wrapping Processes

`smartspawn` also provides functionality to wrap subprocesses, allowing you to modify or set up an environment before execution. Here's how to wrap and unwrap subprocesses:

```
import { startSpawnWrap, endSpawnWrap } from '@push.rocks/smartspawn';

// Start a wrap
startSpawnWrap('path/to/script', ['arg1', 'arg2'], { ENV_VAR: 'value' });
console.log('Subprocess wrapped.');

// End the wrap when it's no longer needed
endSpawnWrap();
console.log('Subprocess unwrapped.');
```

Advanced Thread Management

`@push.rocks/smartspawn` integrates with the `threads` package to provide advanced threading capabilities. Here's a brief example showing how you might utilize it for more complex scenarios:

```
import { Thread, spawn, Worker } from 'threads';

async function advancedThreadExample() {
  const thread = await spawn(new Worker('./worker'));

  const result = await thread.doWork();
  console.log(`Result from thread: ${result}`);

  await Thread.terminate(thread);
}

advancedThreadExample();
```

Ensure you explore the `threads` documentation for a more detailed understanding of creating workers and communicating between the main process and threads.

Conclusion

The `@push.rocks/smartspawn` package simplifies managing subprocesses in your Node.js applications, offering straightforward APIs for starting, stopping, and communicating with child processes. Whether you need to execute a simple script in the background or leverage sophisticated multi-threading capabilities, `smartspawn` provides the tools necessary to implement these features efficiently and robustly.

Remember, the examples above are starting points. Review the source code, tests, and type declarations for a comprehensive understanding of everything `@push.rocks/smartspawn` and its integrated packages can do. Experiment with the library in your projects to best learn how to utilize its full potential in real-world applications.

License and Legal Information

Trademarks

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.