readme.md for @push.rocks/smartsourcemap

get sourcemap support in node

Install

To use `@push.rocks/smartsourcemap` in your project, run:

```
npm install @push.rocks/smartsourcemap --save
```

This will add it to your project's dependencies.

Usage

Getting Started

`@push.rocks/smartsourcemap` makes it straightforward to enable source map support in your Node.js applications. This is particularly useful when working with TypeScript or other compiled languages that produce source maps. By enabling source maps, you can pinpoint the original code position that corresponds to the generated code, making debugging much easier.

Let's dive into how to integrate `@push.rocks/smartsourcemap` into your Node.js or TypeScript project.

Basic Integration

First, ensure that your project is set up with either TypeScript or another compiler that generates source maps. Here's a quick setup in your `tsconfig.json` to enable source map generation:

```
{
  "compilerOptions": {
    "sourceMap": true
  }
}
```

After installing `@push.rocks/smartsourcemap`, integrate it into your application's entry file. For a Node.js project, this might be your `index.js` or the first file that runs when your application starts. For a TypeScript project, this would correspond to your `index.ts` file or its equivalent.

Here's how to integrate `@push.rocks/smartsourcemap` using TypeScript (ESM syntax):

```
import 'source-map-support/register';
```

Or, if you prefer to control when source map support is enabled programmatically, you can do it like this:

```
import { install } from 'source-map-support';
install();
```

This snippet activates the source map support early in your application's lifecycle, ensuring that all subsequent code benefits from accurate source mapping.

Advanced Usage

Although the above method works well for many projects, you might encounter scenarios where you need more control over source map handling. `@push.rocks/smartsourcemap` can be flexibly configured to meet these advanced use cases.

For example, you might want to enable source map support only conditionally, based on environment variables or other runtime conditions. In such cases, you can wrap the installation call in a conditional block, like so:

```
import { install } from 'source-map-support';

if (process.env.NODE_ENV === 'development') {
  install({
    handleUncaughtExceptions: false
  });
}
```

This example demonstrates enabling source map support only in development environments and configuring it not to handle uncaught exceptions, leaving that responsibility to another part of your application.

Integrating with Testing Frameworks

When writing tests for your application, having source map support can be incredibly helpful. It allows you to receive stack traces that point to your original source files rather than the compiled output. Integrating `@push.rocks/smartsourcemap` with testing frameworks like Jest, Mocha, or Tap is straightforward.

Here's an example of enabling source map support in a test setup file when using Jest:

```
// jest.setup.ts
import 'source-map-support/register';
```

And in your `jest.config.js`, reference the setup file:

```
module.exports = {
  setupFilesAfterEnv: ['<rootDir>/jest.setup.ts'],
};
```

Considerations

While `@push.rocks/smartsourcemap` greatly simplifies working with source maps, remember that it operates by rewriting error stack traces. This process can introduce overhead, especially in development environments. In production, it's essential to weigh the benefits of detailed stack traces against any potential performance impacts.

Summary

`@push.rocks/smartsourcemap` provides an essential utility for modern Node.js applications, especially those using TypeScript or other compiled-to-JavaScript languages. By enabling source map support, developers can achieve more accurate debugging, leading to increased productivity and more maintainable codebases. Whether you're building a small tool or a large-scale application, consider integrating `@push.rocks/smartsourcemap` to enhance your development and debugging experience.

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

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 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 11:12:33 UTC by foss.global Team
Updated 2026-03-28 12:19:19 UTC by foss.global Team