@push.rocks/smartping

A utility for performing ping operations in Node.js environments.

- readme.md for @push.rocks/smartping

readme.md for @push.rocks/smartping

a ping utility

Install

To install `@push.rocks/smartping`, run the following command in your project directory:

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

This command adds `@push.rocks/smartping` to your project's dependencies and ensures you can begin utilizing it to manage your network ping needs efficiently.

Usage

`@push.rocks/smartping` leverages TypeScript and ESM syntax for a seamless development experience, offering straightforward methods to conduct ping operations within your applications.

Before diving into the usage scenarios, ensure that you import the module in your TypeScript files using:

```
import { Smartping } from '@push.rocks/smartping';
```

Basic Ping

To perform a basic ping operation to check the reachability of a host, you can do the following:

```
import { Smartping } from '@push.rocks/smartping';

const pingInstance = new Smartping();
```

```
async function basicPing() {
  const pingResponse = await pingInstance.ping('google.com');
  console.log(pingResponse);
}

basicPing();
```

This will output the ping response from `google.com`, including whether the host is alive, the time it took for the response, and other detailed information.

Ping with Timeout

Sometimes, you may want to specify a timeout for the ping operation to avoid long waiting times if the host is not reachable. You can easily do this as follows:

```
import { Smartping } from '@push.rocks/smartping';

const pingInstance = new Smartping();

async function pingWithTimeout() {
  const pingResponse = await pingInstance.ping('google.com', 1000); // Timeout set to 1000
milliseconds
  console.log(pingResponse);
}

pingWithTimeout();
```

Checking if Host is Alive

If you're only interested in whether a host is alive without the need for detailed ping information, you can use the `pingAlive` method:

```
import { Smartping } from '@push.rocks/smartping';

const pingInstance = new Smartping();

async function checkHostAlive() {
  const isAlive = await pingInstance.pingAlive('google.com');
  console.log(`Is Google alive? ${isAlive}`);
```

```
}

checkHostAlive();
```

This method is particularly useful for quickly verifying the availability of a server or an API endpoint.

Advanced Usage Scenarios

`@push.rocks/smartping` can be integrated into health-check mechanisms, automated network diagnostics, server monitoring tools, or any application requiring network communication verification. Its straightforward API and promise-based architecture allow it to be seamlessly incorporated into asynchronous flow control, enhancing both the development experience and performance.

Error Handling

While using `@push.rocks/smartping`, you might encounter errors, particularly when dealing with unreachable hosts or network issues. It is recommended to implement proper error handling to manage such scenarios gracefully:

```
async function safePingWithTimeout() {
  try {
    const pingResponse = await pingInstance.ping('google.com', 500);
    console.log(pingResponse);
  } catch (error) {
    console.error('Ping operation failed:', error);
  }
}

safePingWithTimeout();
```

Wrapping Up

Whether integrating into existing applications for network diagnostics or constructing a new solution requiring ping capabilities, `@push.rocks/smartping` provides an efficient and easy-to-use interface to accomplish these tasks with minimal code. Its design and implementation cater to modern development practices, promoting clean and maintainable code.

For more complex scenarios or contributions, please consult the documentation and source code available on GitHub and NPM. Contributions are always welcome to enhance the module's capabilities and address the evolving needs of developers and applications alike.

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.