@push.rocks/smartopen

A library for opening URLs or files in the default application

readme.md for @push.rocks/smartopen

open things

Install

To start using `@push.rocks/smartopen`, you'll first need to install it using npm. In your terminal, navigate to your project directory and run:

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

This command will add `@push.rocks/smartopen` as a dependency to your project, allowing you to use it in your TypeScript applications.

Usage

Getting Started with SmartOpen

`@push.rocks/smartopen` provides an easy way to programmatically open URLs using Node.js. This can be particularly useful in desktop applications or server-side scripts where you want to trigger opening a web page in the default browser. Below, you'll find comprehensive examples to get you started.

Before diving into the examples, ensure you have TypeScript setup in your project. TypeScript provides strong types for JavaScript, which enhances development through type checking and better tooling support.

Basic Usage

Here is the simplest way to open a URL:

```
import { openUrl } from '@push.rocks/smartopen';

async function openWebPage() {
  await openUrl('https://example.com');
}

openWebPage().then(() => {
  console.log('Web page opened successfully');
}).catch((error) => {
  console.error('Failed to open web page:', error);
});
```

This function imports the `openUrl` method from `@push.rocks/smartopen` and executes it with a URL string. It opens the provided URL in the default web browser.

Advanced Usage: Handling in Continuous Integration Environments

`@push.rocks/smartopen` intelligently detects if it's running in a Continuous Integration (CI) environment and avoids opening the browser in such cases. This is practical for automated testing or deployment scripts where opening a browser window would be unnecessary or problematic.

Here's how you can utilize `@push.rocks/smartopen` in an environment-aware manner:

```
import { openUrl } from '@push.rocks/smartopen';

async function openLinkConditionally(url: string) {
  const result = await openUrl(url);
  if(result === null) {
    console.log('Environment detected as CI, skipping browser opening.');
  } else {
    console.log('Browser should open shortly:', url);
  }
}

openLinkConditionally('https://example.com')
  .catch((error) => console.error('An error occurred:', error));
```

In this example, the `openUrl` function returns `null` if it detects that it's being run in a CI environment, allowing the calling code to handle this scenario appropriately.

Conclusion

`@push.rocks/smartopen` provides a straightforward and effective solution for opening URLs from Node.js scripts, with smart handling for CI environments. Whether you're building a desktop application, developing a server-side application that interacts with web services, or creating automation scripts that open web pages, `@push.rocks/smartopen` offers a reliable method to achieve your goals.

Remember, the examples above demonstrate the basic and advanced usage of `@push.rocks/smartopen`. Depending on your specific needs, you might need to adjust the logic or combine it with other modules to create more complex and powerful 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.

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.