@push.rocks/smartmatch

A minimal matching library using picomatch for string matching based on wildcards.

- readme.md for @push.rocks/smartmatch

readme.md for @push.rocks/smartmatch

a minimal matching library using picomatch

Install

To install `@push.rocks/smartmatch`, you'll need Node.js and npm installed on your development machine. If you have those set up, installing `@push.rocks/smartmatch` is as simple as running the following command in your terminal:

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

This will add `@push.rocks/smartmatch` to your project's dependencies and you're ready to start using it in your application.

Usage

Using `@push.rocks/smartmatch` is straightforward. The library is designed to provide a minimal yet powerful matching functionality leveraging the `picomatch` library. The core functionality revolves around matching strings against a provided wildcard pattern. It’s especially useful for scenarios where there’s a need to filter names, file paths, or any strings based on wildcard patterns (similar to the glob patterns).

Getting Started with @push.rocks/smartmatch

Firstly, ensure that you import `@push.rocks/smartmatch` into your project:

```
import { SmartMatch } from '@push.rocks/smartmatch';
```

Creating a Matcher Instance

The `SmartMatch` class is the entry point to the library's functionality. You create an instance of `SmartMatch` by providing a wildcard pattern to its constructor. This pattern will be used for matching against the strings.

Here’s an example:

```
const myMatcher = new SmartMatch('*.ts');
```

This matcher will be used to determine if a given string matches the wildcard pattern `*.ts`, meaning it should end with `.ts`.

Matching Strings

To match a string against the pattern, use the `match` method on your `SmartMatch` instance. This method takes a string as an argument and returns a boolean indicating whether the string matches the pattern.

Example usage:

```
// Assuming myMatcher is instantiated with '*.ts' as shown above.

const fileName = 'example.ts';

if (myMatcher.match(fileName)) {
  console.log(`${fileName} matches the pattern!`);
} else {
  console.log(`${fileName} does not match the pattern.`);
}
```

Advanced Use Cases

`@push.rocks/smartmatch` is designed to be minimal, but you can create multiple instances of `SmartMatch` with different patterns to support more complex matching logic in your application. This is particularly useful for applications that deal with various types of files, names, or identifiers that follow different naming conventions.

For example:

```
const scriptMatcher = new SmartMatch('*.js');
const typeScriptMatcher = new SmartMatch('*.ts');

const checkFileType = (fileName: string) => {
  if (scriptMatcher.match(fileName)) {
    console.log(`${fileName} is a JavaScript file.`);
  } else if (typeScriptMatcher.match(fileName)) {
    console.log(`${fileName} is a TypeScript file.`);
  } else {
    console.log(`${fileName} is of an unknown type.`);
  }
};

// Test the function
checkFileType('test.js');  // Output: test.js is a JavaScript file.
checkFileType('module.ts');  // Output: module.ts is a TypeScript file.
checkFileType('image.png');  // Output: image.png is of an unknown type.
```

This makes `@push.rocks/smartmatch` a versatile library suitable for various applications that require simple yet efficient matching logic.

Conclusion

`@push.rocks/smartmatch` offers an elegant solution for matching strings against wildcard patterns. Whether you are developing a file utility tool, a project management app, or any software that needs to filter or categorize data based on naming patterns, `@push.rocks/smartmatch` could be the library you need. Its simplicity and effectiveness make it a valuable tool for developers looking for a minimal matching library.

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.