readme.md for @push.rocks/smarti18n

a package dealing with i18n stuff

Install

To use @push.rocks/smarti18n in your project, you will need to install it via npm or yarn. Ensure you have Node.js installed before proceeding. To install the package, run the following command in your project's root directory:

```
npm install @push.rocks/smarti18n --save
```

or if you prefer using yarn:

```
yarn add @push.rocks/smarti18n
```

This command will download @push.rocks/smarti18n and add it to your project's `node_modules` directory as well as add it as a dependency in your project's `package.json` file.

Usage

The `@push.rocks/smarti18n` package provides functionalities to deal with internationalization (i18n) aspects, such as language codes, country codes, and potentially localized strings management. Let's explore its capabilities through various use cases.

Importing the module

First, you need to import the module into your TypeScript project. Since this package uses ESM syntax, you should import it like so:

```
import * as smarti18n from '@push.rocks/smarti18n';
```

Or if you just need specific exports like `countryCodeArray`, you can import them directly:

```
import { countryCodeArray } from '@push.rocks/smarti18n';
```

Working with Country Codes

One primary feature of `@push.rocks/smarti18n` is dealing with country codes. The module provides an array of country codes along with their respective names and phone prefixes.

```
import { countryCodeArray } from '@push.rocks/smarti18n';

// Let's log the name and phone prefix of Germany
const germanyInfo = countryCodeArray.find(country => country.code === 'DE');
console.log(`Country: ${germanyInfo?.name}, Phone Prefix: ${germanyInfo?.phonePrefix}`);
```

This example would output something like:

```
Country: Germany, Phone Prefix: 49
```

This simple scenario showcases how you can retrieve information about a country, which can be particularly useful when working with international forms, user profiles, etc.

Advanced Usage Scenarios

Though the provided test cases and examples primarily deal with country codes, the infrastructure of `@push.rocks/smarti18n` allows for expansions to include handling localized strings based on user locale, formatting numbers and dates according to locale, etc.

Here's a conceptual overview of how you might expand upon the existing framework to include localized strings:

1. **Defining Localized Strings**: Imagine you have JSON files for each locale, containing key-value pairs of strings that need to be localized.

   ```
   // en.json
   {
     "greeting": "Hello, World!"
   }

   // de.json
   ```

```
{
  "greeting": "Hallo, Welt!"
}
```

2. **Loading and Using Localized Strings**: First, you'd load the appropriate JSON file based on the user's locale. Then, you could access the localized strings using a key.

```
import { getLocalizedString } from '@push.rocks/smarti18n';

const userLocale = 'de'; // This would typically come from the user's preferences
const greeting = getLocalizedString('greeting', userLocale);
console.log(greeting); // Outputs: Hallo, Welt!
```

   Note: The function `getLocalizedString` is hypothetical and represents a potential feature of the library, illustrating how it could be expanded to handle a broader scope of i18n needs.

Conclusion

While `@push.rocks/smarti18n` currently focuses on providing a comprehensive listing of country codes, its design allows for easy expansion to cover a wide array of internationalization and localization features. Whether you're building a multinational web application or just need to handle different languages and regions, `@push.rocks/smarti18n` offers a solid foundation to incorporate i18n aspects into your TypeScript projects.

License and Legal Information

Trademarks

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 11:10:43 UTC by foss.global Team
Updated 2026-03-28 12:17:32 UTC by foss.global Team