readme.md for @push.rocks/smartarray

A library providing asynchronous operations like filter, map, and deduplication for arrays in TypeScript.

Install

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

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

Make sure you have Node.js and npm installed beforehand.

Usage

The `@push.rocks/smartarray` library is designed to facilitate asynchronous array operations in TypeScript projects. It simplifies tasks like mapping, filtering, and deduplication by embracing async/await patterns, making it an invaluable tool for modern JavaScript development. Below, we delve into the capabilities of this library, providing comprehensive examples to illustrate its use in a variety of scenarios.

Importing the Library

Before you can utilize the library's functions, you need to import them into your TypeScript files. Depending on your use case, you can import specific functions or the entire library:

```
import { map, filter, deduplicate } from '@push.rocks/smartarray';
```

Async Map: Transforming Arrays

The `map` function lets you apply an asynchronous operation to each item in an array, constructing a new array with the transformed items.

Example: Doubling Numbers

```
const numbers = [1, 2, 3, 4];
const doubleNumbers = await map(numbers, async (number) => number * 2);
console.log(doubleNumbers); // Output: [2, 4, 6, 8]
```

Async Filter: Conditional Array Traversal

With the `filter` function, you can asynchronously judge whether to keep or remove items from the array.

Example: Filtering Even Numbers

```
const numbers = [1, 2, 3, 4, 5, 6];
const evenNumbers = await filter(numbers, async (number) => number % 2 === 0);
console.log(evenNumbers); // Output: [2, 4, 6]
```

Async Deduplicate: Removing Duplication

The `deduplicate` function excels in removing duplicates from an array based on asynchronously derived unique keys for each element.

Example: Deduplicating User Array

```
const users = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 1, name: 'John' }
];
const deduplicatedUsers = await deduplicate(users, async (user) => user.id);
console.log(deduplicatedUsers);
// Output: [{ id: 1, name: 'John' }, { id: 2, name: 'Jane' }]
```

Deep-Dive Use Cases

Complex Data Transformation

Imagine you're working with a dataset of user objects fetched from an API, and you need to perform several transformations: filter out inactive users, double the user IDs for a new report, and ensure the list is deduplicated based on usernames.

```
import { map, filter, deduplicate } from '@push.rocks/smartarray';

// Example users array
const users = [
  { id: 1, active: true, username: 'user1' },
  { id: 2, active: false, username: 'user2' },
  { id: 3, active: true, username: 'user3' },
  { id: 1, active: true, username: 'user1' } // Duplicate for demonstration
];

// First, filter out inactive users
const activeUsers = await filter(users, async (user) => user.active);

// Next, transform the user IDs
const transformedUsers = await map(activeUsers, async (user) => ({
  ...user,
  id: user.id * 2
}));

// Finally, deduplicate based on usernames
const uniqueUsers = await deduplicate(transformedUsers, async (user) => user.username);

console.log(uniqueUsers);
```

This example demonstrates `@push.rocks/smartarray`'s power in handling complex, asynchronous data operations in an efficient, readable manner. By chaining these methods, you can achieve sophisticated data manipulation objectives with minimal code.

Conclusion

`@push.rocks/smartarray` significantly simplifies the development experience when working with arrays in asynchronous environments. It not only enhances readability and maintainability but also ensures that your codebase remains scalable and efficient. By integrating this library into your projects, you unlock a higher level of programming paradigm where array manipulations are no longer a chore but a streamlined process.

For developers aiming to harness the full potential of asynchronous operations in TypeScript, `@push.rocks/smartarray` offers a comprehensive, easy-to-use solution that stands out for its performance and versatility. Whether you're mapping, filtering, or deduplicating arrays, this library empowers you to write cleaner, more efficient code, elevating your development workflow to new heights.

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.

Revision #3
Created 2026-03-28 11:09:58 UTC by foss.global Team
Updated 2026-03-28 12:16:36 UTC by foss.global Team