readme.md for @push.rocks/smartrx

@push.rocks/smartrx

smart wrapper for rxjs

Install

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

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

This package is distributed via npm and should be saved as a dependency in your project's `package.json` file once installed.

Usage

`@push.rocks/smartrx` provides a smart wrapper for working with RxJS, enhancing its already powerful reactive programming capabilities with additional functionalities, including easier observable map management and observable intake handling. We'll explore key features and how to use them in TypeScript.

First, ensure you're working in an environment configured for TypeScript and modern JavaScript development.

Basic Setup

To start using `@push.rocks/smartrx`, first, import what you need from the package:

```
import { Observablemap, ObservableIntake, rxjs } from '@push.rocks/smartrx';
```

Observable Map Management

`Observablemap` helps manage observables efficiently, especially useful when you need to ensure a single observable per event or when working with event emitters.

Basic Observablemap Use

```
import { Observablemap } from '@push.rocks/smartrx';
import { EventEmitter } from 'events';

// Initialize Observablemap
const observableMap = new Observablemap();

// Your event emitter (node.js events in this case)
const myEmitter = new EventEmitter();

// Get a Subject for a specific event
const myEventSubject = observableMap.getSubjectForEmitterEvent(myEmitter, 'myEvent');

// Subscribe to the Subject
myEventSubject.subscribe({
  next: (value) => console.log(`Received value: ${value}`),
});

// Emit events
myEmitter.emit('myEvent', 'Hello World!');
```

This approach ensures that you have a single observable (Subject in this case) per event, efficiently reusing existing observables instead of creating new ones for the same event.

Observable Intake

`ObservableIntake` is designed for efficiently managing and controlling the flow of data through observables, offering features like buffering and intake requests.

Using ObservableIntake

```
import { ObservableIntake } from '@push.rocks/smartrx';

// Initialize ObservableIntake
const observableIntake = new ObservableIntake<string>();

// Listen to the observableIntake as you would with any RxJS Observable
observableIntake.subscribe({
  next: (message) => console.log(message),
  complete: () => console.log('No more messages'),
});

// Push messages into the observable intake
observableIntake.push('Hello');
observableIntake.push('World');

// Signal completion
observableIntake.signalComplete();
```

`ObservableIntake` offers the flexibility of adding values as they come and controlling when those values are emitted to subscribers, including buffering capabilities for managing backpressure.

Advanced Use-cases

`@push.rocks/smartrx` is built to handle more sophisticated scenarios like working with streams or handling events in a web environment.

- **From Streams with Backpressure**: Efficiently create observables from Node.js streams, applying backpressure as needed.
- **Event Management in Browsers**: Easily map browser events to observables, enabling reactive programming principles in frontend development.

Conclusion

`@push.rocks/smartrx` significantly simplifies some of the more tedious aspects of working with RxJS, making it easier to manage observables related to event emitters and providing helpful utilities like observable intake for controlling data flow. With its smart wrappers, developers can focus more on business logic rather than boilerplate code for observable management.

For more complex use cases, such as integrating with external data sources or managing complex state with Redux, `@push.rocks/smartrx` offers a solid foundation for building reactive applications with ease and efficiency.

Remember, reactive programming with RxJS is a powerful paradigm that can make handling asynchronous data streams simpler and more maintainable. `@push.rocks/smartrx` enhances this paradigm by providing tools that make working with RxJS even more pleasant and productive.

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 #2
Created 2026-03-28 11:11:44 UTC by foss.global Team
Updated 2026-03-28 11:41:19 UTC by foss.global Team