# @push.rocks/smartrx

A smart wrapper for rxjs to manage and extend observables.

- readme.md for @push.rocks/smartrx
- docs/index.md for @push.rocks/smartrx
- docs/observablemap.md for @push.rocks/smartrx
- changelog.md for @push.rocks/smartrx

# 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

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

**Please note:** The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

# 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.

# docs/index.md for @push.rocks/smartrx

# smartrx

smart wrapper for rxjs

## Availabililty

npm git git docs

## Status for master

build status coverage report npm downloads per month Dependency Status
bitHound Dependencies bitHound Code TypeScript node JavaScript Style Guide

## Usage

Use TypeScript for best in class instellisense.

For further information read the linked docs at the top of this README.

> MIT licensed | © Lossless GmbH | By using this npm module you agree to our privacy policy

repo-footer

# docs/observablemap.md for @push.rocks/smartrx

# class Observablemap

Event Emitters are nice. However often times you end up with registering multiple listeners for the exact same thing. Observables have a smaller footprint and a more manageable subscribe logic. Observablemap registers only one rxjs observable per event and then continues to reference the same observable that you can subscribe to.

For further information read the linked docs at the top of this README.

# changelog.md for @push.rocks/smartrx

## 2025-04-25 - 3.0.10 - fix(ci)

Update CI workflows, dependency versions, and TypeScript configuration

- Updated .gitea workflow files: changed IMAGE URL and npmci package name (@shipzone/npmci to @ship.zone/npmci)
- Upgraded development dependencies from @gitzone/* to @git.zone/* packages
- Enhanced tsconfig.json with emitDecoratorMetadata, baseUrl, and paths
- Minor code formatting improvements and trailing comma adjustments in TypeScript files
- Updated package.json homepage and added bugs/overrides configuration

## 2025-04-25 - 3.0.9 - fix(deps/ts_plugins)

Update @types/node dependency and adjust rxjs operator exports ordering

- Upgraded @types/node from ^20.8.10 to ^22.15.2 in package.json
- Reordered rxjs operator imports and exports in ts/smartrx.plugins.rxjs.ts for improved clarity

## 2024-05-29 - 3.0.8 - configuration updates

Updated project description and configuration files.

- Updated description.

- Revised tsconfig settings (2024-04-14 and 2024-04-01).
- Updated npmextra.json githost entries (2024-04-01 and 2024-03-30).

# 2023-11-04 - 3.0.7 - core

Fixed core functionality.

- Applied core fixes.

# 2023-11-01 - 3.0.6 - core

Fixed core functionality.

- Applied core fixes.

# 2023-07-24 - 3.0.5 - core

Fixed core functionality.

- Applied core fixes.

# 2023-07-24 - 3.0.4 - core

Fixed core functionality.

- Applied core fixes.

# 2023-07-24 - 3.0.3 - core

Fixed core functionality.

- Applied core fixes.

# 2023-07-12 - 3.0.2 - core

Fixed core functionality and organization scheme.

- Applied core fixes.
- Switched to new organization scheme (commit dated 2023-07-10).

# 2023-06-11 - 3.0.1 - core

Fixed core functionality.

- Applied core fixes.

# 2023-06-10 - 3.0.0 - core

Fixed core functionality.

- Applied core fixes.

---

# 2022-08-05 - 2.0.26 to 2.0.25 - core

Included a breaking change alongside core fixes.

- BREAKING CHANGE: Switched to ESM.
- Applied core fixes.

# 2022-01-24 - 2.0.24 to 2.0.19 - core

Consolidated several minor core fixes.

- Applied multiple core update fixes.

# 2020-09-24 - 2.0.18 - core

Fixed core functionality.

- Applied core fix.

# 2020-07-12 - 2.0.17 - core

Fixed core functionality.

- Applied core fix.

# 2020-06-26 - 2.0.16 to 2.0.15 - core

Fixed core functionality.

- Applied core fixes.

# 2020-05-27 - 2.0.14 to 2.0.6 - core

Fixed core functionality across several releases.

- Applied core fixes.

# 2020-05-26 - 2.0.5 - core

Fixed core functionality.

- Applied core fix.

# 2019-09-10 - 2.0.4 to 2.0.3 - core

Fixed core functionality.

- Applied core fixes.

# 2018-12-10 to 2018-11-23 - 2.0.2 to 2.0.1 - core & dependencies

Fixed core functionality and updated dependencies.

- Applied core fixes (2.0.2).
- Updated dependencies (2.0.1 fix).

# 2018-10-10 - 2.0.0 - core

Fixed core functionality.

- Applied core fix.

---

# 2018-10-10 - 1.0.5 - scope change

Introduced a breaking change in package scope.

- BREAKING CHANGE: Switched to the @pushrocks scope.

# 2017-11-01 - 1.0.4 - testing improvements

Enhanced test quality and intake functionality.

- Improved tests.
- Enabled working intake.

# 2017-10-30 - 1.0.3 - documentation

Updated project documentation.

- Revised docs.

# 2017-10-30 - 1.0.2 - documentation

Enhanced project information.

- Added README.

# 2017-10-26 - 1.0.1 - observable feature

Introduced observability features.

- Added observable functionality.