readme.md for @push.rocks/smartevent

handle events in smart ways

Install

To install @push.rocks/smartevent for use in your project, run the following command in your project root:

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

This command will add `@push.rocks/smartevent` to your project's dependencies, ensuring that its functionality is available for import and use within your application.

Usage

The `@push.rocks/smartevent` package provides an enhanced event handling mechanism on top of the native Node.js `EventEmitter`. It leverages TypeScript and modern JavaScript features to smartly handle events within your application.

Importing the Module

Begin by importing the necessary classes and functions from the module. This can be done as follows:

```
import { SmartEventEmitter, once } from '@push.rocks/smartevent';
```

Using `SmartEventEmitter`

The `SmartEventEmitter` class is an extension of the native `EventEmitter` provided by Node.js but with additional capabilities. Let's see how to use it.

Creating an Instance

```
const mySmartEventEmitter = new SmartEventEmitter();
```

Emitting Events

With an instance of `SmartEventEmitter`, emitting events is straightforward. You simply use the `emit` method.

```
mySmartEventEmitter.emit('myCustomEvent', { key: 'value' });
```

Listening for Events

To listen for events, use the `on` method, passing the event name and a callback function that will be executed when the event is emitted.

```
mySmartEventEmitter.on('myCustomEvent', (data) => {
  console.log('Event data:', data);
});
```

Listening for an Event Once

You might want to listen for an event only once and then automatically remove the listener. This can be achieved with the `once` method.

```
mySmartEventEmitter.once('myCustomEvent', (data) => {
  console.log('This will be logged only once:', data);
});
```

Using `once` Async Function

The `once` function provides a way to wait for an event to occur exactly once before proceeding. It returns a promise that resolves when the specified event is emitted.

```
(async () => {
  const eventData = await once<string>(mySmartEventEmitter, 'myCustomEvent');
  console.log('Event data received asynchronously:', eventData);
})();
```

This functionality is particularly useful when dealing with asynchronous operations that trigger events on completion.

Advanced Use Cases

The simplicity and power of `@push.rocks/smartevent` enable a variety of advanced use cases, such as coordinating complex event-driven flows, or integrating with other promise-based or async/await code seamlessly.

Given its reliance on TypeScript, developers can benefit from strong typing, which adds to code robustness and maintainability. When defining event payloads, consider specifying interfaces or types for the data objects being passed around to ensure type safety across your event listeners.

For those working in environments that support Observables, `@push.rocks/smartevent` can interoperate with libraries like RxJS, allowing events to be transformed into Observable streams for more complex event handling strategies.

Final Thoughts

`@push.rocks/smartevent` is a utilities suite that extends the native event handling capabilities provided by Node.js, giving developers a smarter way to manage events within their applications. Its integration with TypeScript enhances developer experience through better tooling support and type safety. Whether you're building a small utility library or a large-scale application, `@push.rocks/smartevent` provides a solid foundation for your event management needs.

Remember, the most effective way to use `@push.rocks/smartevent` is by thoroughly understanding the events in your application and how they interact with various components. With smart events, you can design a more reactive, maintainable, and scalable application architecture.

For further exploration, the module includes types and interfaces that encourage best practices for event-driven programming in TypeScript, ensuring that you get the most out of your event handling logic.

By integrating `@push.rocks/smartevent` into your project, you empower yourself with a flexible and powerful tool for managing events, which is essential for building responsive and interactive Node.js applications.

License and Legal Information

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.

Revision #3
Created 2026-03-28 11:10:36 UTC by foss.global Team
Updated 2026-03-28 12:17:24 UTC by foss.global Team