# @push.rocks/smartversion

A library to handle semantic versioning with ease.

# readme.md for @push.rocks/smartversion

handle semver with ease

# Install

To incorporate `@push.rocks/smartversion` into your project, run the following command using npm:

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

Or if you prefer using Yarn:

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

This will add it to your project's dependencies.

# Usage

`@push.rocks/smartversion` offers a comprehensive suite of functionalities to easily manipulate and compare semantic versions (semver). The following documentation assumes that you are familiar with TypeScript and semantic versioning concepts.

## Importing the module

Begin by importing `SmartVersion` from the package:

```
import { SmartVersion } from '@push.rocks/smartversion';
```

## Creating a SmartVersion instance

You can instantiate `SmartVersion` with a semver string:

```
const version = new SmartVersion('1.0.0');
console.log(version.versionString); // Outputs: '1.0.0'
```

Alternatively, if you have a fuzzy version string (e.g., `"^1.0.0"`, `"~1.2"`), you can use the static method `fromFuzzyString`:

```
const fuzzyVersion = SmartVersion.fromFuzzyString('^1.0.0');
console.log(fuzzyVersion.versionString); // Outputs the minimum version satisfying the fuzzy string
```

# Accessing Parts of the Version

The major, minor, and patch components are accessible as properties:

```
console.log(version.major); // Outputs: 1
console.log(version.minor); // Outputs: 0
console.log(version.patch); // Outputs: 0
```

# Comparing Versions

You can compare the instance to another `SmartVersion` instance or a semver string to determine if it's greater or lesser:

```
const version1 = new SmartVersion('1.2.3');
const version2 = new SmartVersion('2.0.0');

console.log(version1.greaterThan(version2)); // Outputs: false
console.log(version1.lessThan(version2)); // Outputs: true

console.log(version1.greaterThanString('1.2.2')); // Outputs: true
console.log(version1.lessThanString('2.1.0')); // Outputs: true
```

# Getting a Best Match

For an array of available versions, to find the best match for the given version or range:

```
const availableVersions = ['1.0.0', '1.2.0', '1.2.3', '2.0.0'];
console.log(version.getBestMatch(availableVersions)); // Outputs the best matching version
```

# Getting New Versions

You can also easily increment the version to get a new `SmartVersion` instance of the next major, minor, or patch version:

```
const newPatchVersion = version.getNewPatchVersion();
console.log(newPatchVersion.versionString); // Outputs: '1.0.1'

const newMinorVersion = version.getNewMinorVersion();
console.log(newMinorVersion.versionString); // Outputs: '1.1.0'

const newMajorVersion = version.getNewMajorVersion();
console.log(newMajorVersion.versionString); // Outputs: '2.0.0'
```

# Dynamic Version Updates

For dynamic updates or operations based on conditions:

```
const updateType = 'minor'; // Example condition
const updatedVersion = version.getNewVersion(updateType);
console.log(updatedVersion.versionString); // Outputs: '1.1.0'
```

The `SmartVersion` class and its methods offer a robust solution for managing versions in your projects, enabling you to parse, compare, and manipulate semantic versions programmatically with ease.

This tool is ideal for automated version management in continuous integration / continuous deployment (CI/CD) workflows, package publishing, or anywhere precise version control is needed.

For any updates, contributions, or issues, please visit the GitHub repository or the npm package page.

> Note: This documentation aims to provide comprehensive examples and usage scenarios for `@push.rocks/smartversion`. However, the actual use cases might vary depending on the project context or development environment. It is always recommended to test and validate the functionality within your project setup.

# 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.