@push.rocks/commitinfo

A tool to bake commit information into code for tracking and version control.

readme.md for @push.rocks/commitinfo

bake commitinfos into code

Install

To start using `@push.rocks/commitinfo`, you need to first install it via npm. To do this, run the following command in your terminal:

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

This command will add `@push.rocks/commitinfo` to your project's dependencies in your `package.json` file.

Usage

The `@push.rocks/commitinfo` module is designed to help you incorporate commit information directly into your TypeScript codebase. Below are detailed steps and examples to utilize the module effectively.

Basic Setup

First, ensure that your project is set up to use TypeScript and ESM syntax. Then, you can proceed to import and use `@push.rocks/commitinfo` in your project.

```
import { CommitInfo } from '@push.rocks/commitinfo';

// Specify your project directory and the planned version type of your next commit
const commitInfo = new CommitInfo('./path/to/your/project', 'patch');

// Now you can use commitInfo to access or modify commit-related information
```

Writing Commit Info into Your Project

One of the primary uses of `@push.rocks/commitinfo` is to bake commit information into your project dynamically. This is especially useful for including version information, project description, etc., within your build artifacts.

```
// Assuming you've initialized `commitInfo` as shown previously

// Write commit info into potential directories like 'ts' or 'ts_web'
await commitInfo.writeIntoPotentialDirs(['ts', 'ts_web']);
```

This method will check for the existence of specified directories (e.g., 'ts' and 'ts_web') in your project and create a `00_commitinfo_data.ts` file with commit-related information in each present directory.

Using Commit Info in Your Project

After running the `writeIntoPotentialDirs` function, `00_commitinfo_data.ts` will be generated in the specified directories. You can then import this file anywhere in your project to access the baked commit information.

```
import { commitinfo } from './ts/00_commitinfo_data';

console.log(commitinfo.name); // Outputs: @push.rocks/commitinfo
console.log(commitinfo.version); // Outputs the version that was baked into the code
console.log(commitinfo.description); // Outputs: bake commitinfos into code
```

Real-World Scenario: Automating Version Updates

A practical use case for `@push.rocks/commitinfo` is to automate the updating of version information across your project upon each new release. By integrating `commitInfo.writeIntoPotentialDirs()` into your CI/CD pipeline, you can ensure that each build artifact always contains the latest commit information, making it easier to trace back artifacts to specific versions and commits.

Extending Usage

The above examples provide a basic understanding of how to use `@push.rocks/commitinfo`. However, feel free to explore more advanced scenarios that fit your project's needs, such as customizing the generated `00_commitinfo_data.ts` file or creating a dedicated workflow for handling version bumps and commit information updates.

Conclusion

`@push.rocks/commitinfo` offers a streamlined and efficient way to incorporate dynamic commit information into your TypeScript projects. By following the examples and scenarios outlined above, you can enhance your project’s maintainability and traceability, ensuring that each piece of code can be unequivocally linked to specific commits and version updates.

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.