@push.rocks/smarthash

Provides simplified access to Node.js hash functions, including SHA256 and MD5, with support for strings, streams, and files.

- readme.md for @push.rocks/smarthash
- changelog.md for @push.rocks/smarthash

readme.md for @push.rocks/smarthash

□□ @push.rocks/smarthash

> **Cross-environment hashing made simple** □□
> SHA256 and MD5 hash functions that work seamlessly in both Node.js and browsers, with zero configuration.

npm version TypeScript Cross Platform

□ Why SmartHash?

- □□ **Universal**: Works in Node.js AND browsers without polyfills
- ⚡ **Smart Fallbacks**: Automatically handles Web Crypto API limitations (like non-HTTPS environments)
- □□ **TypeScript First**: Full type safety and IntelliSense support
- □□ **Dual Entry Points**: Optimized builds for both environments
- □□ **Simple API**: Consistent interface across all platforms

□□ Quick Start

```
pnpm install @push.rocks/smarthash
```

□□ API Reference

□□ String Hashing

```
import { sha256FromString, sha256FromStringSync } from '@push.rocks/smarthash';

// Async (works everywhere)
const hash = await sha256FromString('Hello, world!');
console.log(hash); // □□64-character hex string

// Sync (Node.js only)
const hashSync = sha256FromStringSync('Hello, world!');
console.log(hashSync); // ⚡ Instant result
```

□□ File & Stream Hashing (Node.js Only)

```
import { sha256FromFile, sha256FromStream } from '@push.rocks/smarthash';
import fs from 'fs';

// Hash files directly
const fileHash = await sha256FromFile('./myfile.txt');
console.log(fileHash); // □□File's SHA256 hash

// Hash streams (perfect for large files)
const stream = fs.createReadStream('./largefile.zip');
const streamHash = await sha256FromStream(stream);
console.log(streamHash); // □□Stream's SHA256 hash
```

□□ Buffer Hashing

```
import { sha256FromBuffer } from '@push.rocks/smarthash';

// Works with both Buffer (Node.js) and Uint8Array (Browser)
const encoder = new TextEncoder();
const buffer = encoder.encode('Hello, world!');
const bufferHash = await sha256FromBuffer(buffer);
console.log(bufferHash); // □□Buffer's SHA256 hash
```

□□ Object Hashing

```
import { sha256FromObject } from '@push.rocks/smarthash';

// Consistent hashing for JavaScript objects
const myObject = {
  userId: 12345,
  role: 'admin',
  timestamp: Date.now()
};

const objectHash = await sha256FromObject(myObject);
console.log(objectHash); // □□Deterministic object hash
```

> □□ **Pro Tip**: Object property order doesn't matter! `{a: 1, b: 2}` and `{b: 2, a: 1}` produce the same hash.

□□ MD5 Hashing (Node.js Only)

```
import { md5FromString } from '@push.rocks/smarthash';

// Legacy MD5 support (use SHA256 for new projects!)
const md5Hash = await md5FromString('Hello, world!');
console.log(md5Hash); // □□32-character MD5 hash
```

□□ Environment Compatibility

Function	Node.js	Browser	Notes
`sha256FromString`	□	□	Universal support
`sha256FromStringSync`	□	□	Sync not possible in browsers
`sha256FromBuffer`	□	□	Handles Buffer/Uint8Array
`sha256FromFile`	□	□	File system access required

Function	Node.js	Browser	Notes
`sha256FromStream`	□	□	Node.js streams only
`sha256FromObject`	□	□	Uses JSON serialization
`md5FromString`	□	□	Not supported by Web Crypto API

□□ Advanced Usage

Error Handling

```
import { sha256FromString } from '@push.rocks/smarthash';

try {
  const hash = await sha256FromString('sensitive data');
  console.log(`□ Hash computed: ${hash}`);
} catch (error) {
  console.error('□ Hashing failed:', error);
}
```

Browser-Specific Features

In browsers, SmartHash automatically:

- □□ Uses Web Crypto API when available (HTTPS contexts)
- □□ Falls back to pure JavaScript implementation in non-HTTPS environments
- ⚠ Warns when trying to use Node.js-only functions

Import Strategies

```
// Main entry point (Node.js optimized)
import { sha256FromString } from '@push.rocks/smarthash';

// Browser-specific entry point (smaller bundle)
import { sha256FromString } from '@push.rocks/smarthash/web';
```

□□ Development

```
# Run tests (both Node.js and browser)
pnpm test

# Build the project
pnpm build

# Generate documentation
pnpm buildDocs
```

□□ Security Notes

- □ **SHA256**: Cryptographically secure, recommended for all use cases
- ⚠ **MD5**: Legacy support only, not recommended for security-critical applications
- □□ **Cross-Environment**: Produces identical hashes across Node.js and browsers
- □□ **Web Crypto**: Uses native browser APIs when available for maximum performance

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.

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.

changelog.md for @push.rocks/smarthash

2025-09-12 - 3.2.6 - fix(ci)

Add .claude local settings to permit web fetch to www.npmjs.com

- Added .claude/settings.local.json granting WebFetch permission for www.npmjs.com
- Configuration-only change; no runtime or library code modified

2025-09-12 - 3.2.5 - fix(ts_web)

Ensure sha256FromBuffer uses correct ArrayBuffer slice for Uint8Array inputs and add local project config files

- Fix sha256FromBuffer in ts_web: explicitly slice Uint8Array.buffer using byteOffset/byteLength before calling crypto.subtle.digest to ensure the correct ArrayBuffer segment is hashed in browser environments.
- Add .claude/settings.local.json to allow WebFetch to www.npmjs.com for local tooling.
- Add .serena/.gitignore and .serena/project.yml to include project-specific configuration and ignored cache path.
- Add missing newline at end of ts_web/index.ts

2025-09-12 - 3.2.4 - fix(deps)

Bump devDependencies, update smartjson and add workspace/CI settings

- Bump @git.zone/tsbuild from ^2.1.70 to ^2.6.8
- Bump @git.zone/tstest from ^2.3.2 to ^2.3.8
- Bump @push.rocks/smartjson from ^5.0.10 to ^5.2.0
- Add pnpm-workspace.yaml with onlyBuiltDependencies: esbuild, mongodb-memory-server, puppeteer
- Add .claude/settings.local.json to allow WebFetch access to www.npmjs.com

2025-08-03 - 3.2.3 - fix(dependencies/config)

Update dependency versions and add local settings for web fetch configuration

- Bump @git.zone/tstest from ^1.0.81 to ^2.3.2
- Bump @push.rocks/smartenv from ^5.0.5 to ^5.0.13
- Add .claude/settings.local.json to allow WebFetch permissions for www.npmjs.com

2025-08-03 - 3.2.1 - docs(readme)

Enhance readme with comprehensive documentation and modern formatting

- Updated readme.md with enhanced formatting, emojis, and badges
- Added comprehensive API reference with clear examples for all functions
- Included environment compatibility table showing Node.js vs Browser support
- Added advanced usage examples including error handling and import strategies
- Improved documentation structure with better visual organization
- Fixed typo in sha265FromObject examples to sha256FromObject

2025-06-19 - 3.2.0 - feat(package)

Update package.json to use exports field for dual entry points

- Replaced the main and typings fields with an exports object that supports both default and web entry points
- Ensures consistency in module resolution between Node.js and browser environments

2025-06-19 - 3.1.0 - feat(browser)

Implement fallback SHA256 for non-HTTPS environments and enhance browser tests for consistent hashing

- Added a pure JavaScript SHA256 fallback in ts_web/sha256.fallback.ts for environments without crypto.subtle

- Updated ts_web/index.ts to use the fallback when necessary
- Enhanced browser tests in test/test.browser.ts to verify consistent hash outputs
- Reflected new features in documentation updates (readme.plan.md)

2025-06-19 - 3.0.4 - feat

Merge isohash functionality into smarthash to enable cross-environment hash support. This release introduces browser-compatible SHA256 functions via the Web Crypto API and plugins for environment detection and JSON handling.

- Added new plan and implementation steps to merge isohash into smarthash.
- Updated test files to use the new tapbundle import.
- Implemented browser-specific hashing functions in ts_web/index.ts and ts_web/plugins.ts.
- Created browser tests in test/test.browser.ts for SHA256 functions.
- Ensured consistent smarthash functionality across environments.

Note: Several non–breaking maintenance updates (e.g. description, tsconfig, and npmextra.json adjustments) were applied between 2024 and 2023 alongside version marker commits.

2023-09-22 to 2022-06-26 - 3.0.0 - Maintenance

Between versions 3.0.3 and 3.0.0, a series of core fixes and organizational improvements were rolled out.

- Multiple “fix(core)” commits addressed various update needs.
- A couple of releases also switched to a new organization scheme.
- Routine maintenance commits ensured stability across these versions.

2022-06-26 - 2.1.10 - BREAKING CHANGE

A major change was introduced by switching the module system.

- BREAKING CHANGE(core): Switched to ESM, requiring consumers to update their imports accordingly.

2021-03-01 to 2019-11-21 - 2.1.0 - Maintenance

Across versions 2.1.9 down to 2.1.0, the project received multiple fixes and CI updates.

- Repeated “fix(core)” commits improved internal stability.
- A “fix(ci)” update was also introduced to streamline continuous integration processes.

2019-11-21 - 2.0.6 - feat

New functionality was added to expand the available hashing algorithms.

- feat(md5): Now creates MD5 hashes, broadening the project’s cryptographic capabilities.

2019-07-04 to 2018-09-07 - 2.0.0 - Maintenance

This range of releases was dedicated to refining core functionality and enhancing security.

- Numerous “fix(core)” commits ensured consistent behavior.
- A “fix(snyk)” commit added a .snyk file and marked the project as Open Source for improved security auditing.

2018-09-07 - 1.0.4 - BREAKING CHANGE

A breaking change was introduced by renaming the package scope.

- BREAKING CHANGE(scope): Changed the package name to @pushrocks/smarthash, requiring updates for consumers referencing the old name.

2016-08-16 to 2016-05-23 - 1.0.0 - Initial Setup

During the early days of the project, core implementation and structure were established.

- Early commits included the initial implementation (“implementation is ready”), package metadata adjustments (e.g. “update package tags”, “fix README”), and structural additions (“add structure”).
- The journey began with the Initial commit on 2016-05-23, setting the groundwork for future development.