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.

Revision #3
Created 2026-03-28 11:10:41 UTC by foss.global Team
Updated 2026-03-28 12:17:29 UTC by foss.global Team