@push.rocks/webstore

A high-performance storage solution for web applications using IndexedDB.

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

readme.md for @push.rocks/webstore

A high-performance, isomorphic key-value storage solution powered by IndexedDB — works seamlessly in the browser and Node.js with zero config.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Install

Install via npm:

```
npm install @push.rocks/webstore
```

Or with pnpm:

```
pnpm add @push.rocks/webstore
```

Usage

`@push.rocks/webstore` provides a clean, Promise-based API over IndexedDB. It auto-detects the runtime environment — in the browser it uses native IndexedDB, on Node.js it transparently loads `fake-indexeddb` for testing and server-side use.

> All methods are fully typed via TypeScript generics. ESM-only (`"type": "module"`).

□□ Creating a Store

Create a `WebStore` instance by specifying a database name and an object store name:

```
import { WebStore } from '@push.rocks/webstore';

const store = new WebStore<{ name: string; age: number }>({
  dbName: 'myAppDatabase',
  storeName: 'users',
});
```

The generic type parameter `<T>` defines the shape of values stored — giving you full type safety on `get` and `set` operations.

□□ Storing Data

```
await store.set('user:1', { name: 'Alice', age: 30 });
await store.set('user:2', { name: 'Bob', age: 25 });
```

Initialization is automatic — the first call to any data method triggers `init()` behind the scenes, so you don't need to call it manually.

□□ Retrieving Data

```
const user = await store.get('user:1');
console.log(user); // { name: 'Alice', age: 30 }
```

Returns `undefined` if the key doesn't exist.

□ Checking Key Existence

```
const exists = await store.check('user:1');
console.log(exists); // true
```

```
const missing = await store.check('user:999');
console.log(missing); // false
```

□□ Deleting Data

Remove a single entry:

```
await store.delete('user:1');
```

Or clear the entire store:

```
await store.clear();
```

□□ Listing All Keys

```
const allKeys = await store.keys();
console.log(allKeys); // ['user:1', 'user:2']
```

□□ Isomorphic by Design

`WebStore` automatically detects the runtime:

- **Browser** → uses the native `IndexedDB` API directly
- **Node.js** → auto-loads `fake-indexeddb` to provide an in-memory IndexedDB implementation

This means the exact same code works in both environments — perfect for universal/isomorphic apps, SSR frameworks, and testing.

```
// Works identically in Node.js and the browser
const store = new WebStore<string>({
  dbName: 'config',
  storeName: 'settings',
});

await store.set('theme', 'dark');
const theme = await store.get('theme'); // 'dark'
```

⚡ Typed Request Caching

For API-heavy applications, `TypedrequestCache` provides a specialized cache layer for typed requests (compatible with `@api.global/typedrequest-interfaces`). It serializes request method + payload as the cache key and stores the full response.

```
import { TypedrequestCache } from '@push.rocks/webstore';

const cache = new TypedrequestCache('api.example.com');

// Cache a request/response pair
await cache.setByRequest({
  method: 'getUserProfile',
  request: { userId: '123' },
  response: { name: 'Alice', role: 'admin' },
});

// Retrieve cached response by matching method + request
const cached = await cache.getByRequest({
  method: 'getUserProfile',
  request: { userId: '123' },
});

console.log(cached.response); // { name: 'Alice', role: 'admin' }
```

> **Note:** `setByRequest` will throw if the typed request has no `response` field — you can only cache completed request/response pairs.

◫ Lazy Initialization & Deduplication

`WebStore` handles initialization lazily and safely:

- The first data operation automatically triggers `init()`
- Concurrent calls during init are safely queued via a deferred promise
- Calling `init()` explicitly multiple times is safe — subsequent calls are no-ops

```
const store = new WebStore({ dbName: 'mydb', storeName: 'data' });

// Both of these trigger init, but it only runs once
const [a, b] = await Promise.all([
  store.get('key1'),
  store.get('key2'),
]);
```

API Reference

`WebStore<T>`

Method	Signature	Description
`init()`	`() => Promise<void>`	Explicitly initialize the database (called automatically)
`get()`	`(key: string) => Promise<T>`	Retrieve a value by key
`set()`	`(key: string, val: T) => Promise<IDBValidKey>`	Store a value under a key
`check()`	`(key: string) => Promise<boolean>`	Check if a key exists
`delete()`	`(key: string) => Promise<void>`	Delete a single entry
`clear()`	`() => Promise<void>`	Remove all entries from the store
`keys()`	`() => Promise<IDBValidKey[]>`	Get all keys in the store

`TypedrequestCache`

Method	Signature	Description
`setByRequest()`	`(req: ITypedRequest) => Promise<void>`	Cache a completed typed request
`getByRequest()`	`(req: ITypedRequest) => Promise<ITypedRequest>`	Retrieve a cached response by request

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

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 or third parties, 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 or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or 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/webstore

2026-03-26 - 2.0.21 - fix(build)

migrate project tooling and tests to the smartconfig setup and updated IndexedDB dependencies

- replace deprecated npmextra.json metadata with .smartconfig.json
- switch IndexedDB import from @tempfix/idb to idb and add pnpm overrides and patch metadata
- update test imports and rename cross-runtime tests to the node+chromium convention
- add Node.js types and initialize the WebStore database property with a definite assignment assertion

2024-05-29 - 2.0.20 - metadata

Updated project description.

- Refreshed package description metadata
- Includes routine release housekeeping for versions 2.0.19-2.0.20

2024-05-17 - 2.0.14-2.0.19 - core

Applied a series of routine core updates across multiple patch releases.

- Repeated `fix(core): update` changes were released from 2.0.14 through 2.0.19
- Changes appear to be maintenance-level updates with no further detail in commit history

2024-04-14 - 2.0.13 - build

Updated project configuration and package host metadata.

- Updated `tsconfig`
- Updated `npmextra.json` githost settings

2023-05-01 - 2.0.5-2.0.12 - core

Released a sequence of routine core patch updates.

- Repeated `fix(core): update` changes were published from 2.0.5 through 2.0.12
- No additional implementation details were provided in commit messages

2022-05-28 - 2.0.0 - core

Released the 2.0 major version with a breaking module format change.

- Breaking change: switched package output to ESM
- Included routine core update follow-up releases through 2.0.4

2020-10-17 - 1.0.16-1.0.18 - core

Released routine maintenance updates leading up to the final 1.x release.

- Repeated `fix(core): update` changes were published for 1.0.16 and 1.0.17
- Version 1.0.18 introduced the breaking change later released as part of 2.0.0: switch to ESM

2020-07-09 - 1.0.1-1.0.15 - core

Published a long series of routine core maintenance releases.

- Repeated `fix(core): update` changes were released across versions 1.0.1 through 1.0.15
- Commit history does not include more specific change details