@push.rocks/smartstring

A library for handling strings in smart ways, including manipulation and encoding, with TypeScript support.

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

readme.md for @push.rocks/smartstring

Smart string manipulation for TypeScript - Your comprehensive toolkit for handling strings, domains, Git URLs, and encodings with elegance and precision.

Why smartstring?

When working with strings in modern JavaScript/TypeScript applications, you often need more than just basic manipulation. You need to:

- Parse and validate domains and URLs
- Handle Git repository URLs across different formats
- Encode/decode Base64 for data transmission
- Normalize indentation in code generators
- Create cryptographically secure random strings
- Validate string encodings
- Parse Docker environment variables

smartstring unifies all these capabilities in a single, tree-shakeable, TypeScript-native package with zero setup overhead.

Install

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

Or using pnpm (recommended):

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

Features at a Glance

- **Domain parsing** - Extract protocols, subdomains, domains, and TLDs
- **Git URL handling** - Parse and convert between SSH and HTTPS formats
- **Base64 encoding** - Standard and URL-safe Base64 operations
- **Smart indentation** - Indent, outdent, and normalize multi-line strings
- **Random strings** - Pattern-based and cryptographically secure generation
- **String normalization** - Clean and standardize whitespace
- **Type checking** - Validate UTF-8 and Base64 encodings
- **Docker env parsing** - Transform Docker environment arrays to objects

Usage

Domain Parsing

Extract detailed information from any URL or domain string:

```
import { Domain } from '@push.rocks/smartstring';

const domain = new Domain('https://subdomain.example.com:3000/path');

console.log(domain.protocol);        // 'https'
console.log(domain.hostname);        // 'subdomain.example.com'
console.log(domain.port);            // '3000'
console.log(domain.pathname);        // '/path'
console.log(domain.fullUrl);         // 'https://subdomain.example.com:3000/path'

// Domain level extraction
console.log(domain.level1);          // 'com' - TLD
console.log(domain.level2);          // 'example' - Domain
console.log(domain.level3);          // 'subdomain' - Subdomain
console.log(domain.zoneName);        // 'example.com' - Full domain without subdomain
console.log(domain.subdomain);       // 'subdomain'
```

Git Repository URLs

Parse and convert Git repository URLs between SSH and HTTPS formats:

```
import { GitRepo } from '@push.rocks/smartstring';

// Parse SSH format
const repo = new GitRepo('git@github.com:user/awesome-project.git');
console.log(repo.host);              // 'github.com'
console.log(repo.user);              // 'user'
console.log(repo.repo);              // 'awesome-project'
console.log(repo.httpsUrl);          // 'https://github.com/user/awesome-project.git'

// Parse HTTPS format
const httpsRepo = new GitRepo('https://gitlab.com/team/project.git');
console.log(httpsRepo.sshUrl);       // 'git@gitlab.com:team/project.git'
console.log(httpsRepo.httpsUrl);     // 'https://gitlab.com/team/project.git'
```

□□ Base64 Encoding

Handle Base64 encoding with both standard and URL-safe variants:

```
import { Base64, base64 } from '@push.rocks/smartstring';

// Using the Base64 class
const encoded = new Base64('Hello World! □□', 'string');
console.log(encoded.base64String);      // Standard Base64
console.log(encoded.base64UriString);   // URL-safe Base64
console.log(encoded.simpleString);      // Decoded string

// Using utility functions
const quickEncode = base64.encode('Secret message');
const quickDecode = base64.decode(quickEncode);

// Validate Base64 strings
console.log(base64.isBase64('SGVsbG8='));           // true
console.log(base64.isBase64('Not Base64!'));        // false

// URL-safe Base64 operations
const urlSafeEncoded = base64.encodeUri('https://example.com/path?param=value');
const urlSafeDecoded = base64.decodeUri(urlSafeEncoded);
```

□□ Smart Indentation

Manage indentation in multi-line strings with precision:

```
import { indent } from '@push.rocks/smartstring';

// Indent with spaces
const indented = indent.indent('Line 1\nLine 2\nLine 3', 4);
// Result:
//     Line 1
//     Line 2
//     Line 3

// Indent with custom prefix
const prefixed = indent.indentWithPrefix('Item 1\nItem 2', '> ');
// Result:
// > Item 1
// > Item 2

// Add prefix to first line only
const firstLinePrefixed = indent.addPrefix('Chapter\nContent here', '# ');
// Result:
// # Chapter
// Content here

// Normalize irregular indentation
const messy = `
    function hello() {
            console.log('Hi');
        return true;
    }
`;
const clean = indent.normalize(messy);
// Result: Properly aligned with minimum indentation preserved
```

□□ Random String Generation

Create random strings with specific patterns or cryptographic security:

```
import { create } from '@push.rocks/smartstring';

// Pattern-based random strings
// A = uppercase, a = lowercase, 0 = number, ! = special, * = any
const password = create.createRandomString('Aa0!', 16);
// Example: "Kg7$Lp2@Qm9#Xn4!"

const alphanumeric = create.createRandomString('Aa0', 10);
// Example: "K7gLp2Qm9X"

const numbers = create.createRandomString('0', 6);
// Example: "472819"

// Cryptographically secure random strings
const cryptoId = create.createCryptoRandomString();
// Example: "f7b2d8e0-3c4a-4b9c-8d2e-1f0a7b9c8d7e"
```

□□ String Normalization

Clean and standardize strings for consistent formatting:

```
import { normalize } from '@push.rocks/smartstring';

const messyString = `
    This text has
        inconsistent indentation
            and too much whitespace



    between lines...
`;

const cleaned = normalize.standard(messyString);
// Result: Properly formatted with consistent spacing

// Custom normalization
const customNormalized = normalize.normal(messyString);
```

□□ String Type Validation

Check string encodings and types:

```
import { type } from '@push.rocks/smartstring';

// Check if string is valid UTF-8
const isValidUtf8 = type.isUtf8('Hello □□');
console.log(isValidUtf8);  // true

// Check if string is Base64
const isBase64String = type.isBase64('SGVsbG8gV29ybGQ=');
console.log(isBase64String);  // true
```

□□ Docker Environment Variables

Parse Docker-style environment variable arrays:

```
import { docker } from '@push.rocks/smartstring';

const envArray = [
  'NODE_ENV=production',
  'PORT=3000',
  'DATABASE_URL=postgresql://localhost:5432/mydb',
  'API_KEY=secret123'
];

const envObject = docker.makeEnvObject(envArray);

console.log(envObject.NODE_ENV);      // 'production'
console.log(envObject.PORT);          // '3000'
console.log(envObject.DATABASE_URL);  // 'postgresql://localhost:5432/mydb'
console.log(envObject.API_KEY);       // 'secret123'

// Use in Docker-related configurations
// Perfect for parsing process.env or Docker Compose outputs
```

API Reference

Classes

- **`Domain`** - URL/domain parser with component extraction
- **`GitRepo`** - Git repository URL parser and converter
- **`Base64`** - Base64 encoder/decoder with multiple formats

Modules

- **`create`** - Random string generation
  - `createRandomString(pattern, length, options)` - Pattern-based generation
  - `createCryptoRandomString()` - Cryptographically secure strings
- **`indent`** - Indentation management
  - `indent(text, spaces)` - Add spaces to each line
  - `indentWithPrefix(text, prefix)` - Add custom prefix to each line
  - `normalize(text)` - Fix inconsistent indentation
  - `addPrefix(text, prefix)` - Add prefix to first line only
- **`normalize`** - String normalization
  - `standard(text)` - Apply standard normalization
  - `normal(text)` - Basic normalization
- **`type`** - String type checking
  - `isUtf8(text)` - Validate UTF-8 encoding
  - `isBase64(text)` - Validate Base64 format
- **`base64`** - Base64 utilities
  - `encode(text)` - Standard Base64 encoding
  - `decode(text)` - Standard Base64 decoding
  - `encodeUri(text)` - URL-safe Base64 encoding
  - `decodeUri(text)` - URL-safe Base64 decoding
  - `isBase64(text)` - Check if string is valid Base64
- **`docker`** - Docker utilities
  - `makeEnvObject(envArray)` - Convert env array to object

Real-World Examples

Building a URL Shortener

```
import { Domain, create, base64 } from '@push.rocks/smartstring';

function createShortUrl(longUrl: string): string {
  const domain = new Domain(longUrl);
  const shortCode = create.createRandomString('Aa0', 6);
  const encoded = base64.encodeUri(`${domain.hostname}${domain.pathname}`);

  return `short.ly/${shortCode}`;
}
```

Processing Code Templates

```
import { indent, normalize } from '@push.rocks/smartstring';

function generateComponent(name: string, props: string[]): string {
  const propsSection = props
    .map(prop => `${prop}: string;`)
    .join('\n');

  const template = `
export interface ${name}Props {
${indent.indent(propsSection, 2)}
}

export function ${name}(props: ${name}Props) {
${indent.indent('// Component implementation', 2)}
}
`;

  return normalize.standard(template);
}
```

Git Repository Manager

```
import { GitRepo } from '@push.rocks/smartstring';

class RepoManager {
```

```
  repos: Map<string, GitRepo> = new Map();

  addRepo(url: string): void {
    const repo = new GitRepo(url);
    this.repos.set(repo.repo, repo);
  }

  getCloneCommand(name: string, useSSH = false): string {
    const repo = this.repos.get(name);
    if (!repo) throw new Error('Repository not found');

    const url = useSSH ? repo.sshUrl : repo.httpsUrl;
    return `git clone ${url}`;
  }
}
```

Browser Support

This package is built for modern environments and includes:

- □ Full ES Module support
- □ Tree-shaking ready
- □ TypeScript definitions
- □ Browser-compatible (via bundlers)
- □ Node.js 14+ support

Development

```
# Clone the repository
git clone https://code.foss.global/push.rocks/smartstring.git

# Install dependencies
pnpm install

# Run tests
pnpm test
```

```
# Build the project
pnpm build
```

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.

changelog.md for @push.rocks/smartstring

2025-09-12 - 4.1.0 - feat(core)

Introduce native implementations for Base64, random generation and normalization; remove runtime plugin dependencies; update tests, docs and package metadata

- Implemented a cross-platform universal Base64 encoder/decoder and replaced usage of external js-base64 with the internal implementation
- Added a custom, cross-platform random string generator (uses crypto.getRandomValues when available) and removed dependency on randomatic / crypto-random-string
- Replaced strip-indent usage with an internal stripIndent implementation in normalize utilities
- Switched domain parsing to use the standard URL class instead of external url.parse
- Simplified ts/smartstring.plugins.ts to only export @push.rocks/isounique and removed several plugin re-exports (js-base64, strip-indent, randomatic, url, smartenv)
- Updated test imports to use @git.zone/tstest/tapbundle
- Expanded and updated README with full usage, examples and API reference
- Updated package.json: trimmed dependencies, bumped @git.zone/tstest version, added packageManager (pnpm) entry and adjusted files/browserslist
- Added .claude/settings.local.json

2024-05-29 - 4.0.15 - maintenance

Package metadata and TypeScript configuration updates.

- Updated package description.
- TypeScript configuration (tsconfig) adjustments.
- npmextra.json updated (githost entries).

2024-03-03 - 4.0.14 → 4.0.1 - maintenance & core fixes

Routine core fixes, small updates and version bumps across the 4.0.x series.

- Multiple "fix(core): update" commits addressing internal/core issues.
- Several version-only releases (patch bumps and releases with minimal change notes).

2022-03-18 - 4.0.2 → 3.0.26 - release consolidation

Consolidated releases and small fixes spanning late 2020–early 2022.

- Several patch releases with minor fixes and version bumps.
- Routine maintenance and stability improvements.

2018-11-28 - 3.0.4 → 3.0.0 - CI & dependency fixes

Improvements to build/CI and dependency cleanups for the 3.0.x line.

- fix(dependencies and structure): updated dependencies and project structure (3.0.4).
- fix(ci): reduced build dependencies and fixed build steps (3.0.3, 3.0.2).
- fix(dependencies): updated test framework and removed obsolete dependency on typings-global (3.0.1, 3.0.0).
- Numerous small fixes and version bump releases across 3.0.x.

2018-07-21 - 2.0.28 - BREAKING CHANGE: package scope

Breaking change: package scope changed.

- Changed package scope to @pushrocks (BREAKING CHANGE).

2017-10-26 - 2.0.27 → 2.0.25 - code quality & npm metadata

Small refactors and npm metadata additions.

- Refactor to use const (2.0.27).
- Added npmextra.json (2.0.26).
- Added create module (2.0.25).

2016-10-31 - 2.0.19 → 2.0.17 - Base64 & exports

Added Base64 handling and improved exports.

- Added Base64 handling and exposed base64 functions separate from class (2.0.17–2.0.19).
- Small export fixes.

2016-07-17 - 2.0.14 → 2.0.12 - ES6, indent/deindent, tests

Feature enhancements around string indentation and ES6 migration.

- Switched codebase to ES6 (2.0.14).
- Implemented deindent and working indent module; added tests (2.0.12).
- Prep work for indent functionality and recompiled builds (2.0.11, 2.0.10).

2016-06-21 - 2.0.9 → 2.0.6 - domain handling improvements

Domain parsing and regex improvements.

- Fixed Domain regex to include '-' and '_' and improved fullName handling (2.0.9, 2.0.8).
- Now evaluates Domains without protocol specified (2.0.6).

2016-05-25 - 2.0.4 → 2.0.0 - core features & CI/docs

Major 2.0.0 work and related improvements.

- Now computes zoneName and detects protocol (2.0.0).
- Added authors note, improved README and CI (AppVeyor, GitLab CI) across several commits.
- Multiple small fixes, dependency updates and YML/CI tweaks (2.0.4, 2.0.3, 2.0.2, 2.0.1).

2016-05-16 - 1.0.3 → 1.0.1 - parser & typings fixes

Parser improvements and TypeScript declaration fixes.

- Correctly parsing a Domain and structure updates (1.0.3).
- Fixes for typings and declaration issues (1.0.2, 1.0.1).

2016-04-15 - 1.0.0 → 0.0.3 - initial stable release

Initial stable release artifacts.

- Added TypeScript regex section and performed first 1.0.0 release (1.0.0).
- Prior 0.0.x preparatory releases.

2016-02-23 - 0.0.0 → initial - project initialization

Project initialization and first commits.

- Initial commit and early CI/travis updates.