@push.rocks/smartfeed

A library for creating and parsing various feed formats.

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

readme.md for @push.rocks/smartfeed

The modern TypeScript library for creating and parsing RSS, Atom, and Podcast feeds □□

`@push.rocks/smartfeed` is a powerful, type-safe feed management library that makes creating and parsing RSS 2.0, Atom 1.0, JSON Feed, and Podcast feeds ridiculously easy. Built with TypeScript from the ground up, it offers comprehensive validation, security features, and supports modern podcast standards including iTunes tags and the Podcast namespace.

Features □

- □□ **Full TypeScript Support** - Complete type definitions for all feed formats
- □□ **Cross-Platform** - Works in Node.js, Bun, Deno, and browsers
- □□ **Multiple Feed Formats** - RSS 2.0, Atom 1.0, JSON Feed 1.0, and Podcast RSS
- □□ **Modern Podcast Support** - iTunes tags, Podcast 2.0 namespace (guid, medium, locked, persons, transcripts, funding)
- □□ **Built-in Validation** - Comprehensive validation for URLs, emails, domains, and timestamps
- □□ **Security First** - XSS prevention, content sanitization, and secure defaults
- □□ **Zero Config** - Works out of the box with sensible defaults
- □□ **Feed Parsing** - Parse existing RSS and Atom feeds from strings or URLs
- □□ **Flexible API** - Create feeds from scratch or from standardized article arrays

Installation

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

Quick Start

Creating a Basic Feed

```
import { Smartfeed } from '@push.rocks/smartfeed';

const smartfeed = new Smartfeed();

// Create a feed
const feed = smartfeed.createFeed({
  domain: 'example.com',
  title: 'Tech Insights',
  description: 'Latest insights in technology and innovation',
  category: 'Technology',
  company: 'Example Inc',
  companyEmail: 'hello@example.com',
  companyDomain: 'https://example.com'
});

// Add an item
feed.addItem({
  title: 'TypeScript 5.0 Released',
  timestamp: Date.now(),
  url: 'https://example.com/posts/typescript-5',
  authorName: 'Jane Developer',
  imageUrl: 'https://example.com/images/typescript.jpg',
  content: 'TypeScript 5.0 brings exciting new features...'
});

// Export as RSS, Atom, or JSON
const rss = feed.exportRssFeedString();
const atom = feed.exportAtomFeed();
const json = feed.exportJsonFeed();
```

Custom Feed URL

You can specify a custom URL for your feed's self-reference instead of the default `https://${domain}/feed.xml`:

```
const feed = smartfeed.createFeed({
  domain: 'example.com',
  title: 'My Blog',
  description: 'Latest posts',
```

```
  category: 'Technology',
  company: 'Example Inc',
  companyEmail: 'hello@example.com',
  companyDomain: 'https://example.com',
  feedUrl: 'https://cdn.example.com/feeds/main.xml' // Custom feed URL
});

// The feedUrl will be used in:
// - RSS: <atom:link href="..." rel="self">
// - Atom: <link href="..." rel="self">
// - JSON Feed: "feed_url" field
```

This is particularly useful when your feed is hosted on a CDN or different domain than your main site.

Creating a Podcast Feed

```
import { Smartfeed } from '@push.rocks/smartfeed';

const smartfeed = new Smartfeed();

const podcast = smartfeed.createPodcastFeed({
  domain: 'podcast.example.com',
  title: 'The Tech Show',
  description: 'Weekly discussions about technology',
  category: 'Technology',
  company: 'Tech Media Inc',
  companyEmail: 'podcast@example.com',
  companyDomain: 'https://example.com',
  // iTunes tags
  itunesCategory: 'Technology',
  itunesAuthor: 'John Host',
  itunesOwner: {
    name: 'John Host',
    email: 'john@example.com'
  },
  itunesImage: 'https://example.com/artwork.jpg',
  itunesExplicit: false,
  itunesType: 'episodic',
```

```
  // Podcast 2.0 tags
  podcastGuid: '92f49cf0-db3e-5c17-8f11-9c5bd9e1f7ec', // Permanent GUID
  podcastMedium: 'podcast', // or 'music', 'video', 'film', 'audiobook', 'newsletter', 'blog'
  podcastLocked: true, // Prevent unauthorized imports
  podcastLockOwner: 'john@example.com'
});

// Add an episode
podcast.addEpisode({
  title: 'Episode 42: The Future of AI',
  authorName: 'John Host',
  imageUrl: 'https://example.com/episode42.jpg',
  timestamp: Date.now(),
  url: 'https://example.com/episodes/42',
  content: 'In this episode, we explore the future of artificial intelligence...',
  audioUrl: 'https://example.com/audio/episode42.mp3',
  audioType: 'audio/mpeg',
  audioLength: 45678900, // bytes
  itunesDuration: 3600, // seconds
  itunesEpisode: 42,
  itunesSeason: 2,
  itunesEpisodeType: 'full',
  itunesExplicit: false,
  // Modern podcast features
  persons: [
    { name: 'John Host', role: 'host' },
    { name: 'Jane Guest', role: 'guest', href: 'https://example.com/jane' }
  ],
  transcripts: [
    { url: 'https://example.com/transcripts/ep42.txt', type: 'text/plain' }
  ],
  funding: [
    { url: 'https://example.com/support', message: 'Support the show!' }
  ]
});

// Export podcast RSS with iTunes and Podcast namespace
const podcastRss = podcast.exportPodcastRss();
```

Parsing Existing Feeds

```
import { Smartfeed } from '@push.rocks/smartfeed';

const smartfeed = new Smartfeed();

// Parse from URL
const feed = await smartfeed.parseFeedFromUrl('https://example.com/feed.xml');
console.log(feed.title);
console.log(feed.items.map(item => item.title));

// Parse from string
const xmlString = '<rss>...</rss>';
const parsedFeed = await smartfeed.parseFeedFromString(xmlString);
```

Creating Feeds from Article Arrays

```
import { Smartfeed } from '@push.rocks/smartfeed';
import type { IArticle } from '@tsclass/tsclass';

const smartfeed = new Smartfeed();

const articles: IArticle[] = [
  // Your article objects conforming to @tsclass/tsclass IArticle interface
];

const feedOptions = {
  domain: 'blog.example.com',
  title: 'My Blog',
  description: 'Thoughts on code and design',
  category: 'Programming',
  company: 'Example Inc',
  companyEmail: 'blog@example.com',
  companyDomain: 'https://example.com'
};

// Creates an Atom feed from articles
```

```
const atomFeed = await smartfeed.createFeedFromArticleArray(feedOptions, articles);
```

API Reference

Smartfeed Class

The main class for creating and parsing feeds.

`createFeed(options: IFeedOptions): Feed`

Creates a standard feed (RSS/Atom/JSON).

Options:

- `domain` (string) - Feed domain (e.g., 'example.com')
- `title` (string) - Feed title
- `description` (string) - Feed description
- `category` (string) - Feed category
- `company` (string) - Company/organization name
- `companyEmail` (string) - Contact email
- `companyDomain` (string) - Company website URL (absolute)
- `feedUrl` (string, optional) - Custom URL for the feed's self-reference (defaults to `https://${domain}/feed.xml`)

`createPodcastFeed(options: IPodcastFeedOptions): PodcastFeed`

Creates a podcast feed with iTunes and Podcast namespace support.

iTunes Options:

- `itunesCategory` (string) - iTunes category
- `itunesSubcategory` (string, optional) - iTunes subcategory
- `itunesAuthor` (string) - Podcast author
- `itunesOwner` (object) - Owner info with `name` and `email`
- `itunesImage` (string) - Artwork URL (1400x1400 to 3000x3000, JPG/PNG)
- `itunesExplicit` (boolean) - Explicit content flag
- `itunesType` ('episodic' | 'serial', optional) - Podcast type
- `itunesSummary` (string, optional) - Detailed summary
- `copyright` (string, optional) - Custom copyright
- `language` (string, optional) - Language code (default: 'en')

Podcast 2.0 Options:

- `podcastGuid` (string) - **Required.** Globally unique identifier (GUID) for the podcast
- `podcastMedium` ('podcast' | 'music' | 'video' | 'film' | 'audiobook' | 'newsletter' | 'blog', optional) - Content medium type
- `podcastLocked` (boolean, optional) - Prevents unauthorized podcast imports (e.g., to other platforms)
- `podcastLockOwner` (string, optional) - Email of who can unlock (required if `podcastLocked` is true)

`parseFeedFromUrl(url: string): Promise<ParsedFeed>`

Parses an RSS or Atom feed from a URL.

`parseFeedFromString(xmlString: string): Promise<ParsedFeed>`

Parses an RSS or Atom feed from an XML string.

`createFeedFromArticleArray(options: IFeedOptions, articles: IArticle[]): Promise<string>`

Creates an Atom feed from an array of `@tsclass/tsclass` article objects.

Feed Class

Represents a feed that can be exported in multiple formats.

`addItem(item: IFeedItem): void`

Adds an item to the feed.

Item Properties:

- `title` (string) - Item title
- `timestamp` (number) - Unix timestamp in milliseconds
- `url` (string) - Absolute URL to the item
- `authorName` (string) - Author name
- `imageUrl` (string) - Absolute URL to featured image
- `content` (string) - Item content/description
- `id` (string, optional) - Unique identifier (uses URL if not provided)

`exportRssFeedString(): string`

Exports the feed as RSS 2.0 XML.

`exportAtomFeed(): string`

Exports the feed as Atom 1.0 XML.

`exportJsonFeed(): string`

Exports the feed as JSON Feed 1.0.

PodcastFeed Class

Extends `Feed` with podcast-specific functionality.

`addEpisode(episode: IPodcastItem): void`

Adds a podcast episode to the feed.

Episode Properties (in addition to IFeedItem):

- `audioUrl` (string) - Absolute URL to audio file
- `audioType` (string) - MIME type (e.g., 'audio/mpeg')
- `audioLength` (number) - File size in bytes
- `itunesDuration` (number) - Duration in seconds
- `itunesEpisode` (number, optional) - Episode number
- `itunesSeason` (number, optional) - Season number
- `itunesEpisodeType` ('full' | 'trailer' | 'bonus', optional)
- `itunesExplicit` (boolean, optional) - Explicit content flag
- `itunesSubtitle` (string, optional) - Short description
- `itunesSummary` (string, optional) - Detailed summary
- `persons` (array, optional) - People involved (hosts, guests)
- `chapters` (array, optional) - Chapter markers
- `transcripts` (array, optional) - Transcript links
- `funding` (array, optional) - Donation/support links

`exportPodcastRss(): string`

Exports the podcast feed as RSS 2.0 with iTunes and Podcast namespace extensions.

Validation & Security

`@push.rocks/smartfeed` includes comprehensive validation to ensure feed integrity and security:

- **URL Validation** - All URLs must be absolute and use http/https protocols
- **Email Validation** - Email addresses are validated against RFC standards

- **Domain Validation** - Proper domain format checking
- **Timestamp Validation** - Ensures timestamps are valid and reasonable
- **Content Sanitization** - Prevents XSS attacks through proper XML escaping
- **Duplicate Detection** - Prevents duplicate item IDs in feeds
- **Required Field Checking** - Validates all required fields are present

Best Practices

Feed Item IDs

Feed item IDs should be permanent and never change once published. This allows feed readers to properly track which items have been read:

```
feed.addItem({
  id: 'post-2024-01-15-typescript-tips', // Permanent ID
  title: 'TypeScript Tips',
  url: 'https://example.com/posts/typescript-tips',
  // ... other fields
});
```

If you don't provide an `id`, the `url` will be used. Make sure URLs don't change for published items.

HTTPS URLs

Always use HTTPS URLs for security and privacy. The library will warn you if HTTP URLs are used:

```
// □ Good
imageUrl: 'https://example.com/image.jpg'

// ⚠□ Will trigger a warning
imageUrl: 'http://example.com/image.jpg'
```

Podcast Artwork

For podcast feeds, artwork should be:

- Square (1:1 aspect ratio)

- Between 1400x1400 and 3000x3000 pixels
- JPG or PNG format
- Maximum 512 KB file size (Apple Podcasts requirement)

Podcast 2.0 Compatibility

The library fully supports the Podcast 2.0 namespace, making your feeds compatible with modern podcast platforms like:

- **Podcast Index** - The open podcast directory
- **Castopod** - Open-source podcast hosting platform
- **Podverse** - Open-source podcast app
- And other Podcast 2.0-compliant apps

Key Podcast 2.0 Features:

- `podcast:guid` - Permanent unique identifier for your podcast
- `podcast:medium` - Declare if your feed is a podcast, music, video, etc.
- `podcast:locked` - Protect your podcast from unauthorized imports
- `podcast:person` - List hosts, co-hosts, and guests with rich metadata
- `podcast:transcript` - Link to transcript files in various formats
- `podcast:funding` - Add donation/support links for your listeners

These features are included in the RSS export when you use `exportPodcastRss()`.

TypeScript Support

Full TypeScript definitions are included. Import types as needed:

```
import type {
  IFeedOptions,
  IFeedItem,
  IPodcastFeedOptions,
  IPodcastItem,
  IPodcastOwner,
  IPodcastPerson,
  IPodcastChapter,
  IPodcastTranscript,
  IPodcastFunding
} from '@push.rocks/smartfeed';
```

Why @push.rocks/smartfeed?

- **Type-Safe** - Catch errors at compile time, not runtime
- **Modern Standards** - Full support for latest podcast specifications
- **Secure by Default** - Built-in validation and sanitization
- **Developer Friendly** - Intuitive API with great error messages
- **Well Tested** - Comprehensive test suite ensuring reliability
- **Actively Maintained** - Regular updates and improvements

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.

changelog.md for @push.rocks/smartfeed

2025-11-01 - 1.4.0 - feat(feed)

Support custom feedUrl for feeds and use it as the self-link in RSS/Atom/JSON; update docs

- Add optional feedUrl option to IFeedOptions (ts/classes.feed.ts).
- Use feedUrl as the atom:link rel="self" href in RSS and as the in Atom when provided.
- Expose feedUrl as the JSON Feed "feed_url" value (ts/classes.feed.ts).
- PodcastFeed now uses podcastOptions.feedUrl for its atom self-link (ts/classes.podcast.ts).
- Update README: add a Custom Feed URL section and mention feedUrl in the API docs (readme.md).

2025-10-31 - 1.3.0 - feat(parsing)

Replace rss-parser with fast-xml-parser and add native feed parser; update Smartfeed to use parseFeedXML and adjust plugins/tests

- Replaced dependency on rss-parser with fast-xml-parser (package.json + deno.lock).
- Added ts/lib/feedparser.ts: a new native XML-based feed parser that detects and parses RSS 2.0, Atom 1.0 and RSS 1.0 (RDF) into a unified IParsedFeed structure.
- Updated Smartfeed parsing API to use parseFeedXML for parseFeedFromString and to fetch + parse XML in parseFeedFromUrl.
- Updated ts/plugins.ts to export fast-xml-parser's XMLParser instead of rss-parser.
- Implemented feed parsing utilities: content extraction, snippet creation, date normalization, enclosure/category handling and atom:link/feed metadata extraction.
- Added and/or updated comprehensive tests for creation, export, parsing, validation, podcast (Podcast 2.0) features to exercise the new parser and related behaviors.

2025-10-31 - 1.2.0 - feat(podcast)

Add Podcast 2.0 support and remove external 'feed' dependency; implement internal RSS/Atom/JSON generators and update tests/README

- Add Podcast 2.0 fields and validation: podcastGuid (required), podcastMedium, podcastLocked and podcastLockOwner
- Include Podcast 2.0 tags in RSS export (podcast:guid, podcast:medium, podcast:locked, podcast:person, podcast:transcript, podcast:funding)
- Remove dependency on the external 'feed' package and replace with internal feed generation for RSS, Atom and JSON Feed
- Update ts/plugins.ts to stop exporting the removed 'feed' plugin
- Update numerous tests to provide podcastGuid and exercise Podcast 2.0 features
- Documentation updated (readme.md) to document Podcast 2.0 support and examples

2025-10-31 - 1.1.1 - fix(podcast)

Improve podcast episode validation, make Feed.itemIds protected, expand README and add tests

- PodcastFeed.addEpisode: validate iTunes duration separately (require itunesDuration) and ensure it is a positive number; audioLength must be a positive number; moved itunesDuration out of the generic required-fields list to allow proper numeric validation and clearer errors.
- Feed: changed itemIds from private to protected so subclasses (e.g. PodcastFeed) can access and enforce duplicate ID checks across episodes/items.
- Documentation: major README overhaul with Quick Start, Podcast examples, API reference, validation & security notes, best practices, and TypeScript usage examples.
- Tests: added comprehensive podcast tests (advanced features and validation) and updated/expanded test coverage for feed creation, export, parsing and validation to cover transcripts, funding, persons, explicit flags, and more.
- This is a backwards-compatible bugfix and documentation/test update; no breaking public API changes intended.

2025-10-31 - 1.1.0 - feat(smartfeed)

Implement Smartfeed core: add feed validation, parsing, exporting and comprehensive tests

- Implement Feed class with full option validation, addItem validation (URLs, email, timestamp), duplicate ID protection, content sanitization and generation of RSS/Atom/JSON feeds.

- Add validation utilities (validateUrl, validateDomain, validateEmail, validateTimestamp, validateRequiredFields, sanitizeContent) in ts/validation.ts used across the module.
- Implement Smartfeed class functions: createFeed, createFeedFromArticleArray, parseFeedFromString and parseFeedFromUrl with rss-parser integration.
- Adjust module exports (ts/index.ts) and plugin imports (ts/plugins.ts) to match implemented classes.
- Add comprehensive test suite under test/ (creation, export, parsing, validation, integration) to exercise new functionality.
- Add deno.lock to lock dependency graph for reproducible builds.

2025-10-31 - 1.0.11 - smartfeed / feed

Add feed and validation utilities for the smartfeed plugin and perform related dependency, refactor, test, and CI updates.

- feat: implement feed and validation utilities for smartfeed to support improved feed generation and input validation.
- chore: bump feed dependency to v5.1.0 and adjust import paths for consistency with the updated package.
- refactor: improve Feed and SmartFeed class structure and formatting for readability and maintainability.
- fix: update test imports to use the new package path after refactor/import changes.
- chore: streamline plugin exports to a consistent structure.
- chore: update README for clarity and formatting improvements.
- chore: update TypeScript configuration for better compatibility.
- ci: add workflows to handle tag and non-tag pushes.

2020-10-25 to 2024-05-29 - 1.0.1..1.0.11 - housekeeping

Collection of minor releases, metadata updates and routine fixes made across multiple intermediate versions.

- Multiple small "fix(core): update" changes and routine release markers (1.0.2 → 1.0.11).
- Updates to package metadata and npmextra.json (githost) across several commits.
- Switch to new organization naming/scheme.
- Miscellaneous tsconfig and description updates.

- These changes were primarily maintenance, CI/package metadata, and release housekeeping.