Basic-Markdown-To-PDF

A lightweight TypeScript library that converts Markdown files into styled PDF documents. Built on marked for parsing and PDFKit for PDF generation.

Features

- **Headings** (h1–h6) with configurable fonts, sizes, and colors
- **Inline formatting** — bold, italic, bold-italic, strikethrough, inline code
- **Code blocks** with monospace font and background shading
- **Blockquotes** with colored left border
- **Lists** — ordered and unordered, with nested sub-lists
- **Tables** with header row highlighting and cell borders
- **Links** rendered as clickable PDF hyperlinks
- **Images** — local (PNG, JPEG) and remote (HTTP/HTTPS) with automatic SVG-to-PNG conversion via @resvg/resvg-js
- **Horizontal rules**
- **Automatic page breaks** when content exceeds the current page
- **Fully themeable** — customize fonts, colors, spacing, page size, and margins

Installation

```
npm install
```

Quick Start

CLI

Convert a Markdown file to PDF from the command line:

```
npx ts-node src/cli.ts <input.md> [output.pdf]
```

output path is omitted, the PDF is written alongside the input file with a `.pdf` extension.

```
# Converts README.md !’ README.pdf
npx ts-node src/cli.ts README.md

# Explicit output path
npx ts-node src/cli.ts docs/report.md output/report.pdf
```

Programmatic API

```
import { generatePdf } from './src/index';

// File-based — reads Markdown from disk, writes PDF to disk
await generatePdf('samples/sample.md', 'output/sample.pdf');

// Buffer-based — returns a PDF Buffer for further processing
import { renderMarkdownToPdf } from './src/index';

const markdown = '# Hello World\n\nThis is a **test**.';
const pdfBuffer = await renderMarkdownToPdf(markdown);
```

Generate Sample PDFs

`samples/` directory contains example Markdown files. Generate PDFs for all of them at once:

```
npm run generate
```

written to the `output/` directory.

Configuration

`renderMarkdownToPdf` accept an optional `PdfOptions` object:

```
interface PdfOptions {
  theme?: ThemeConfig;     // Typography, colors, and component styles
  pageLayout?: PageLayout; // Page size and margins
  basePath?: string;       // Base directory for resolving relative image paths
}
```

Page Layout

```
import { generatePdf } from './src/index';

await generatePdf('input.md', 'output.pdf', {
  pageLayout: {
    pageSize: 'A4',
    margins: { top: 72, right: 72, bottom: 72, left: 72 },
  },
});
```

The default layout uses **Letter** page size with 50pt margins on all sides.

Theming

Override any part of the default theme to customize the look of the generated PDF:

```
import { generatePdf, defaultTheme } from './src/index';

await generatePdf('input.md', 'output.pdf', {
  theme: {
    ...defaultTheme,
    headings: {
      ...defaultTheme.headings,
      h1: { font: 'Helvetica-Bold', fontSize: 32, color: '#0a3d62', bold: true },
    },
    linkColor: '#e74c3c',
  },
});
```

`ThemeConfig` interface exposes styles for:

Section	Configurable properties
`headings`	Font, size, and color for each level (h1–h6)
`body`	Font, size, color, and line gap
`code.inline`	Font, size, color, and background color
`code.block`	Font, size, color, background color, and padding
`blockquote`	Border color, border width, italic flag, and indent
`table`	Header background, border color, and cell padding
`linkColor`	Color for hyperlink text
`horizontalRuleColor`	Color for `---` dividers

Project Structure

```
% % %  src/
%    % % %  index.ts      # Public API — generatePdf, renderMarkdownToPdf, exports
%    % % %  cli.ts        # Command-line entry point
%    % % %  renderer.ts   # Markdown-to-PDF rendering engine
%    % % %  styles.ts     # Default theme and page layout
%    % % %  types.ts      # TypeScript interfaces for options and theming
% % %  samples/
%    % % %  generate.ts   # Script to batch-generate sample PDFs
%    % % %  sample.md     # Full-featured sample document
%    % % %  image.md      # Image rendering tests (local, remote, SVG, broken)
%    % % %  logo.svg      # Sample SVG image
%    % % %  logo.png      # Sample PNG image
%    % % %  sample.png    # Sample raster image
% % %  output/           # Generated PDF output directory
% % %  package.json
% % %  tsconfig.json
```

Dependencies

Package	Purpose
[marked](https://github.com/markedjs/marked)	Markdown parsing and tokenization
[pdfkit](https://pdfkit.org/)	PDF document generation
[@resvg/resvg-js](https://github.com/nicolo-ribaudo/resvg-js)	SVG-to-PNG rasterization for image embedding

Scripts

Command	Description
`npm run build`	Compile TypeScript to `dist/`
`npm run generate`	Generate sample PDFs from `samples/*.md` into `output/`

License

ISC