

# Smoke Test

## TABLE OF CONTENTS

[venomx-pdf-cli](#)

[Features](#)

[Quick start](#)

[CLI usage](#)

[Custom CSS tips](#)

[Development](#)

[Functionality + Logic Flow](#)

[System functionality](#)

[Internal logic flow](#)

[Core logic pseudo-code](#)

## venomx-pdf-cli

Convert Markdown files into a polished, print-ready PDF with one command. Built-in themes, syntax highlighting, optional Table of Contents, and custom CSS keep exports consistent and offline-friendly.

## Features

- Merge one or many `.md` files into a single PDF (order preserved)
- Built-in themes: `default` , `dark` , `clean`
- Syntax highlighting for code blocks (`highlight.js`)
- Optional Table of Contents ( `--toc` ) generated from headings
- Custom document title ( `--title` ) and custom CSS overrides ( `--css` )
- Works fully offline once dependencies are installed; suited for CI/CD pipelines

## Quick start

```
npm install
npm run build
node dist/index.js README.md -o README.pdf --theme dark --toc
```

## CLI usage

```
vpdf <files...> [options]
# alias
venomx <files...> [options]
```

### Options:

- `-o, --output <file>` Output PDF path (default: first file name with `.pdf`)
- `--theme <name>` Theme to apply ( `default` , `dark` , `clean` )
- `--theme <path>` Provide a CSS file as the theme
- `--toc` Add a Table of Contents based on document headings
- `--title <title>` Override the document title
- `--css <file>` Append a custom CSS file after the chosen theme
- `--html-only` Emit HTML only (skip PDF generation)
- `--no-sandbox` Launch Chromium without sandboxing (useful in restricted CI)
- `--splash` Show a welcome splash screen and exit (can be run without files)

### Examples:

```
# Merge multiple files with a TOC and dark theme
vpdf intro.md api.md guide.md --toc --theme dark -o docs.pdf
# same with alias
venomx intro.md api.md guide.md --toc --theme dark -o docs.pdf

# Add custom title and CSS overrides
vpdf README.md --title "Project Handbook" --css styles/custom.css
# alias
venomx README.md --title "Project Handbook" --css styles/custom.css
```

## Custom CSS tips

Your CSS is appended after the selected theme. A simple override:

```
:root {  
  --accent: #f97316;  
  --font-body: "Georgia", serif;  
}  
.doc-title { text-align: center; }
```

## Development


- `npm run dev` Run the CLI with ts-node
- `npm run build` Emit compiled files to `dist/`
- `npm run clean` Remove the build output
- `npm test` Build then run a fast HTML snapshot test (no Chromium launch)

## Functionality + Logic Flow

### System functionality

- Markdown → PDF conversion: merge one or many `.md` files into a single PDF with consistent formatting.
- Theme support: built-in `default`, `dark`, `clean` plus optional custom CSS via `--css ./my-style.css`.
- Table of Contents: auto-generates clickable links from document headings when `--toc` is provided.
- Syntax highlighting: code blocks are highlighted with `highlight.js`.
- Output customization: `-o output.pdf`, `--title "My Project Docs"`, A4 page size by default.
- CLI execution: `vpdf input.md` (installed) or `npx venomx-pdf-cli`.

### Internal logic flow

1. Parse CLI inputs with commander; validate file paths, output name, theme name, flags ( `--toc` , `--title` , `--css` , `--no-sandbox` ).
2. Read Markdown files with `fs.readFile` ; if multiple, preserve order and label sections.
3. Convert Markdown → HTML via `markdown-it` (HTML enabled, linkify, typographer) + `highlight.js` ; collect heading text/levels.
4. Generate TOC when enabled: normalize heading levels (H1–H6) and build anchor links.
5. Load theme CSS: apply built-in theme and append custom CSS if provided.
6. Build final HTML document: inject styles, TOC (if any), and rendered sections.
7. Render PDF with Puppeteer: headless Chromium loads HTML and exports PDF ( `printBackground` , A4).
8. Save and finish: write PDF to disk and log  PDF created: `<path>` .

## Core logic pseudo-code

```
function runConversion(options) {
  const files = resolveFiles(options.input);
  const slugCounts = new Map();
  const md = createMarkdownRenderer(slugCounts);

  const sections = files.map((file) => {
    const content = readFile(file);
    const { html, headings } = renderMarkdown(md, content, slugCounts);
    return { html, headings, label: basename(file) };
  });

  const toc = options.toc ? generateToc(flattenHeadings(sections)) : null;
  const themeCss = loadTheme(options.theme) + loadCustomCss(options.css);
  const html = wrapHtml({ title: options.title, toc, sections, themeCss });

  exportPdf(html, options.output, options.noSandbox);
}
```

