MD2PDF - Beautiful Markdown to PDF Converter

A comprehensive Python tool that converts Markdown files to impeccably beautiful PDFs with professional styling, syntax highlighting, and responsive layout. Features a dynamic style and theme system with automatic discovery and a powerful workflow for batch processing.

> 🔒 **Team Access Only**: This package is distributed privately via GitHub releases for MPS Metalmind AB team members.

✨ Features

🎨 **Dynamic Style System**: Beautiful typography templates with automatic discovery from CSS files
🌈 **Color Theme Engine**: Sophisticated color themes dynamically loaded from theme files
📝 **Advanced Markdown Support**: Full syntax highlighting, tables, TOC, footnotes
📄 **Professional Layout**: Optimized for A4 paper with proper margins and page breaks
🔄 **Batch Processing**: Default workflow for processing entire folders
🎯 **Code Block Management**: Proper wrapping and syntax highlighting
📊 **Table Support**: Beautifully formatted tables with alternating row colors
🔗 **Link Styling**: Elegant link formatting with hover effects
📋 **List Formatting**: Properly styled ordered and unordered lists
💬 **Blockquotes**: Distinguished quote styling with left border
🖼️ **Image Support**: Responsive image handling with rounded corners
📖 **Table of Contents**: Automatic TOC generation for better navigation
😀 **Emoji-Safe Rendering**: Twemoji SVG replacement for robust PDF output (local assets preferred, CDN fallback)
🖍️ **Reliable Syntax Highlighting**: Embedded Pygments CSS, improved comment contrast per theme, and neutralized error tokens to avoid red boxes around ASCII art

🚀 Quick Start

Installation

From GitHub Release (Team Access):

```
# Install the latest release wheel directly
pip install https://github.com/mps-metalmind/md2pdf/releases/latest/download/md2pdf-1.0.0-py3-none-any.whl

# Or install from a specific version
pip install https://github.com/mps-metalmind/md2pdf/releases/download/v1.0.0/md2pdf-1.0.0-py3-none-any.whl
```

From Repository (Development):

```
# Install directly from the repository
pip install git+https://github.com/mps-metalmind/md2pdf.git

# Or install a specific version/tag
pip install git+https://github.com/mps-metalmind/md2pdf.git@v1.0.0
```

Local Development Installation:

```
git clone https://github.com/mps-metalmind/md2pdf.git
cd md2pdf
pip install -e ".[dev,docs,test]"
```

System Dependencies

MD2PDF requires some system libraries for PDF generation:

macOS:

```
brew install cairo pango gdk-pixbuf libffi
```

Ubuntu/Debian:

```
sudo apt-get install build-essential python3-dev python3-pip \
  python3-setuptools python3-wheel python3-cffi libcairo2 \
  libpango-1.0-0 libpangocairo-1.0-0 libgdk-pixbuf2.0-0 \
  libffi-dev shared-mime-info
```

Windows:

```
# Install Visual Studio Build Tools
# WeasyPrint will handle most dependencies automatically
```

Basic Usage

Command Line Interface:

```
# Single file conversion
md2pdf document.md
md2pdf document.md --style modern --theme elegant
md2pdf document.md --output custom_output.pdf

# With custom header
md2pdf document.md --header header.md
md2pdf document.md --header /path/to/header/directory/

# Batch processing
md2pdf *.md --style technical --theme dark
md2pdf "docs/*.md" --style story --theme sepia

# List available options (dynamically discovered)
md2pdf list-styles
```

Python API:

```python
from md2pdf import MD2PDFConverter

# Simple conversion
converter = MD2PDFConverter()
converter.convert('document.md', 'output.pdf')

# With custom styling
converter = MD2PDFConverter(
    style='modern',
    theme='elegant'
)
converter.convert('document.md', 'styled_output.pdf')

# Batch processing
converter.convert_batch('*.md', output_dir='pdfs/')
```

Project Structure

```
MD2PDF/
├── md2pdf.py            # PDF converter entry point
├── md2word.py           # Word converter entry point
├── requirements.txt     # Python dependencies
├── quick_start.sh       # Quick setup script
├── README.md            # This file
├── .gitignore           # Git ignore rules
├──
├── src/                 # Source code
│   ├── converters/      # Document converters
│   │   ├── base_converter.py
│   │   ├── pdf_converter.py
│   │   └── word_converter.py
│   ├── processors/      # Processing modules
│   │   ├── markdown_processor.py
│   │   ├── header_processor.py
│   │   └── workflow_processor.py
│   ├── utils/           # Utilities
│   │   └── style_loader.py
│   └── main/            # Main entry points
│       ├── md2pdf.py
│       └── md2word.py
├──
├── data/                # Data directories
│   ├── input/           # Source markdown files (workflow)
│   ├── output/          # Generated documents (workflow)
│   ├── processed/       # Processed files (workflow)
│   └── header/          # Default header assets (can be overridden with --header option)
├──
├── assets/              # Static assets (fonts, emojis)
├── styles/              # Style templates (CSS)
│   └── templates/       # Style CSS files (dynamically discovered)
│       ├── technical.css  # Technical documentation
│       ├── modern.css     # Modern, sophisticated
│       ├── whitepaper.css # Academic, authoritative
│       ├── story.css      # Literary, elegant
│       ├── academic.css   # Formal, scholarly
│       ├── consultancy.css # Business consulting
│       └── futuristic.css # Bold futuristic design
├── themes/              # Color themes (dynamically discovered)
│   ├── default.css      # Clean and professional
│   ├── minimal.css      # Sophisticated, timeless
│   ├── sophisticated.css  # Refined light design
│   ├── elegant.css      # Sophisticated dark design
│   ├── dark.css         # Dark containers
│   ├── midnight.css     # Dark containers with contrast
│   ├── oceanic.css      # Cool, calming blue tones
│   ├── forest.css       # Natural, earthy green palette
│   ├── sepia.css        # Warm, vintage colors
│   └── agile.css        # Agile/Scrum themed colors
├──
├── scripts/             # Utility scripts
│   ├── example.py       # Usage examples
│   └── temp_scripts/    # Temporary/debug scripts
```

```
    ├──
    ├── docs/              # Documentation
    │   ├── CLAUDE.md
    │   └── samples/         # Sample outputs
    └──
    └── venv/              # Virtual environment
```

🎨 Style Templates

Available Styles

Styles are dynamically discovered from CSS files in the `styles/templates/` directory. Run `md2pdf list-styles` to see all currently available styles.

Common styles include:

STYLE	DESCRIPTION	BEST FOR
Technical	Clean, professional, code-friendly	Technical documentation, APIs, guides
Modern	Sophisticated, elegant, contemporary	Premium documentation, presentations
Whitepaper	Elegant, academic, authoritative	Research papers, business documents
Story	Literary, elegant, readable	Creative writing, narratives
Academic	Formal, scholarly, citation-friendly	Research papers, theses
Consultancy	Business consulting presentation style	Business reports, consulting docs
Futuristic	Bold futuristic design	Modern tech docs, presentations

Style Features

- **Typography**: Each style uses carefully selected fonts (Inter, JetBrains Mono, etc.)
- **Layout**: Optimized spacing, margins, and typography hierarchy
- **Code Blocks**: Proper syntax highlighting with theme-appropriate backgrounds
- **Print Optimization**: A4 page layout with proper page breaks
- **Responsive Design**: Adapts to different content types

🌈 Color Themes

Available Themes

Themes are dynamically discovered from CSS files in the `themes/` directory. Run `md2pdf list-styles` to see all currently available themes.

Common themes include:

THEME	TYPE	DESCRIPTION
Default	Light	Clean and professional
Minimal	Light	Sophisticated, elegant, timeless
Sophisticated	Light	Refined light design with subtle accents
Elegant	Dark	Sophisticated dark design
Dark	Light	Dark containers with light text
Midnight	Light	Dark containers with high contrast
Oceanic	Light	Cool, calming blue tones
Forest	Light	Natural, earthy green palette
Sepia	Light	Warm, vintage book-like colors
Agile	Light	Agile/Scrum themed colors

Theme Features

- **CSS Custom Properties**: Dynamic theming with CSS variables
- **Print Optimization**: All themes designed for PDF output
- **Code Block Styling**: Proper contrast for syntax highlighting
- **Consistent Design**: Unified color palette across all elements

🔄 Workflow System

Default Workflow

The default workflow processes all markdown files in the `input/` folder:

1. **Place files**: Add `.md` files to the `input/` folder
2. **Run converter**: `python md2pdf.py --style <style> --theme <theme>`
3. **Generated PDFs**: Appear in `output/` folder with naming: `{filename}_{style}_{theme}.pdf`
4. **Processed files**: Original files moved to `processed/` folder

Workflow Features

- **Automatic folder creation**: Creates `input/`, `output/`, `processed/` if missing
- **Duplicate handling**: Appends numbers to avoid filename conflicts
- **Batch processing**: Converts all files in one command
- **Progress tracking**: Shows conversion status and results

Example Workflow

```
# 1. Add files to input folder
cp my_document.md input/

# 2. Run workflow
python md2pdf.py --style modern --theme sophisticated

# 3. Check results
ls output/        # Generated PDFs
ls processed/     # Original files
```

📖 Advanced Usage

Command Line Options

```
md2pdf [INPUT] [OPTIONS]

Arguments:
  INPUT                 Input markdown file or glob pattern

Options:
  -o, --output PATH       Output PDF file or directory
  -s, --style STYLE       Style template (default: technical, run 'md2pdf list-styles' to see all)
  -t, --theme THEME       Color theme (default: default, run 'md2pdf list-styles' to see all)
  --header PATH           Path to header markdown file or directory with header content
  --output-dir PATH       Output directory for batch processing
  --verbose               Enable verbose output
  --help                  Show help message

Commands:
  convert                 Convert markdown files (default command)
  list-styles             List all available styles and themes
  batch                   Batch convert files in a directory
```

Python API Reference

```python
from md2pdf import MD2PDFConverter, StyleManager

# Initialize converter
converter = MD2PDFConverter(
    style='modern',          # Style template
    theme='elegant',          # Color theme
    output_dir='pdfs/',      # Output directory
    verbose=True             # Enable logging
)

# Convert single file
converter.convert(
    input_path='document.md',
    output_path='document.pdf'  # Optional
)

# Convert multiple files
converter.convert_batch(
    pattern='*.md',
    output_dir='output/'
)

# Get conversion metadata
metadata = converter.get_metadata('document.md')
print(f"Title: {metadata['title']}")
print(f"Pages: {metadata['page_count']}")
```

Style and Theme Management

```python
from md2pdf import StyleManager

style_manager = StyleManager()

# List available options
styles = style_manager.list_styles()
for style in styles:
    print(f"{style.name}: {style.description}")

themes = style_manager.list_themes()
for theme in themes:
    print(f"{theme.name}: {theme.description}")

# Get all valid combinations
combinations = style_manager.get_combinations()
for style, theme in combinations:
    print(f"{style} + {theme}")

# Load custom styles
style_manager.load_custom_style('path/to/custom.css')
```

🛠️ Customization

Adding New Styles

1. Create a new CSS file in `styles/templates/` folder
2. Add descriptive comment at the top: `/* Style Name - Description */`
3. Define CSS variables for theming
4. Style will be automatically discovered and available in the CLI

   Example:

```
/* Custom Style - My Special Style */
@import url('https://fonts.googleapis.com/css2?family=...');

:root {
    --font-body: 'Your Font', sans-serif;
    --font-heading: 'Your Heading Font', serif;
    --font-code: 'Your Code Font', monospace;
    /* ... other variables */
}

/* ... rest of your CSS ... */
```

Adding New Themes

1. Create a new CSS file in `themes/` folder
2. Add descriptive comment at the top: `/* Theme Name - Description */`
3. Define theme CSS variables
4. Theme will be automatically discovered

Example:

```
/* Custom Theme - My Special Theme */
:root {
    --theme-primary: #your-color;
    --theme-secondary: #your-color;
    --theme-text: #your-color;
    --theme-background: #your-color;
    --theme-surface: #your-color;
    --theme-code-bg: #your-color;
    --theme-code-text: #your-color;
    /* ... other theme variables */
}
```

📋 Requirements

- **Python**: 3.8+
- **System Dependencies**: Cairo, Pango, GDK-Pixbuf (see installation instructions)
- **Python Dependencies**:

- `markdown>=3.5.0` - Markdown processing with extensions
- `weasyprint>=58.0` - High-quality PDF generation
- `pygments>=2.15.0` - Syntax highlighting
- `beautifulsoup4>=4.12.0` - HTML processing
- `click>=8.0.0` - Command-line interface
- `rich>=13.0.0` - Rich terminal output
- `pydantic>=2.0.0` - Data validation
- `PyYAML>=6.0` - YAML configuration
- `Pillow>=10.0.0` - Image processing

🧪 Aesthetic Considerations

Typography

- High-legibility body fonts and clear monospace code fonts per style (e.g., Inter, Source Serif Pro, JetBrains Mono).
- Heading weights differ by style (lighter in Modern, stronger in Technical/Whitepaper) to signal hierarchy.

Spacing & Padding

- A4 page margins defined via `@page`; dark themes use a `.content` container with inner padding to provide breathing room.
- For multi-page documents, container padding is cloned across page fragments to maintain consistent spacing.

First-line Indents

- Narrative styles (Story, Academic) indent paragraphs for readability but never the first paragraph after a heading.
- Technical, Modern, and Whitepaper avoid first-line indents by default.

Text Alignment

- Body text alignment varies by style (left vs. justified), but list items are always ragged-right (left-aligned) to avoid rivers and awkward spacing.
- Emojis render as inline images and are aligned to baseline to prevent line breaks.

Code & Syntax Highlighting

- Class-based Pygments with embedded CSS ensures consistent rendering across themes.
- Comment tokens have tuned contrast per theme; code blocks and inline code include background fills.
- Error token styling is neutralized to avoid red boxes around ASCII art/trees.

🎯 Examples

Technical Documentation

```
python md2pdf.py api_docs.md --style technical --theme dark
```

Academic Paper

```
python md2pdf.py research_paper.md --style academic --theme sophisticated
```

Creative Writing

```
python md2pdf.py short_story.md --style story --theme sepia
```

Business Document

```
python md2pdf.py whitepaper.md --style whitepaper --theme minimal
```

Batch Processing

```
# Process all documentation
python md2pdf.py --style modern --theme elegant
```

🔧 Troubleshooting

Common Issues

WeasyPrint errors: Ensure all system dependencies are installed

```
# macOS
brew install cairo pango gdk-pixbuf libffi

# Ubuntu/Debian
sudo apt-get install build-essential python3-dev python3-pip python3-setuptools python3-wheel python3-cffi li
bcairo2 libpango-1.0-0 libpangocairo-1.0-0 libgdk-pixbuf2.0-0 libffi-dev shared-mime-info
```

Font issues: Ensure Google Fonts are accessible or use system fonts
Code block wrapping: All styles include proper text wrapping for long lines

Debug Mode

For troubleshooting, you can inspect the generated HTML:

```
converter = MD2PDFConverter('document.md')
html_content = converter._process_markdown(content)
print(html_content)  # Inspect the HTML output
```

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🤝 Contributing

We welcome contributions! Please see our Contributing Guidelines for details.

Quick Contributing Steps:

1. Fork the repository
2. Create a feature branch: `git checkout -b feature/amazing-feature`
3. Install development dependencies: `pip install -e ".[dev,test]"`
4. Make your changes and add tests
5. Run the test suite: `pytest`
6. Run code quality checks: `pre-commit run --all-files`
7. Commit your changes: `git commit -m 'Add amazing feature'`
8. Push to your branch: `git push origin feature/amazing-feature`
9. Submit a pull request

Adding Custom Styles and Themes

- **Styles**: Add CSS files to `src/md2pdf/styles/templates/`
- **Themes**: Add CSS files to `src/md2pdf/themes/`
- Files are automatically discovered - no registration needed
- Follow the existing naming conventions and documentation standards
- Include descriptive comments at the top of each file

📞 Support

- **Documentation**: https://md2pdf.readthedocs.io
- **Issues**: GitHub Issues
- **Discussions**: GitHub Discussions
- **Email**: info@metalmind.se

MD2PDF - Transform your markdown into beautiful, professional PDFs with style and elegance. ✨