INT. DIGITAL WORKSPACE - DAY

Screen recording shows markdown text transforming into a professional screenplay PDF.

NARRATOR
(excited, urgent)
What if I told you that every screenplay you've ever seen follows an incredibly strict format...

Quick cuts of professional screenplays from Final Draft and industry scripts flash across the screen.

NARRATOR
...and that we built a tool to automate it entirely using AI pair programming?

Markdown syntax highlighting appears, then transforms smoothly into a formatted PDF.

CUT TO:

INT. STUDIO - DAY

TITLE CARD

The title "Building ScriptMD2PDF with Claude Code" animates onto screen.

NARRATOR
(confident)
Today I'm showing you how we built ScriptMD2PDF with Claude Code—and why this changes everything about AI-assisted development.

Claude Code interface and terminal windows appear on screen.

CUT TO:

INT. STUDIO - CONTINUOUS

NARRATOR
(conversational)
Here's the thing about writing screenplays: the formatting is brutal.

SPLIT SCREEN

Complex screenplay software Final Draft on the left, ruler measurements on the right.

Md2Script-Script

NARRATOR
You've got scene headings at 1.5 inches from the left. Dialogue at 2.5 inches. Character names at 3.5 inches.

Animated rulers appear showing exact measurements.

NARRATOR
And if you get it wrong? Your script looks immediately unprofessional.

NARRATOR
(frustrated tone)
Traditional solutions? Either expensive software like Final Draft that costs hundreds of dollars—

Final Draft pricing page appears on screen.

NARRATOR
—or complex LaTeX templates that require a computer science degree to configure.

LaTeX code with complicated syntax fills the screen.

NARRATOR
(solution tone)
But what if you just want to write in markdown—the simplest, most portable format on earth—and get a perfectly formatted screenplay PDF out the other end?

A clean, simple markdown editor with screenplay text appears.

NARRATOR
That's exactly what we built. And we did it with Claude Code as our AI pair programmer.

CUT TO:

INT. DIGITAL WORKSPACE - DAY

SIDE-BY-SIDE VIEW

Markdown source on the left, rendered PDF on the right, synchronized scrolling.

Md2Script-Script

NARRATOR
(enthusiastic)
ScriptMD2PDF is a Python-powered conversion tool that takes screenplay-flavored markdown and outputs production-ready PDFs that follow industry-standard formatting.

NARRATOR
Here's what the syntax looks like:

CLOSE ON
Code example with syntax highlighting appears:

INT. KITCHEN - DAY

The morning light filters through venetian blinds.

ALEX
(whispering)
Did you hear that?

JORDAN
We need to leave. Now.

CUT TO:

NARRATOR
(clear, instructional)
Three hash marks for scene headings. At-symbol for character names. Double arrows for transitions. That's it.

LIVE DEMO
Typing markdown, instant PDF preview updating in real-time.

NARRATOR
The tool handles all the heavy lifting: proper margins, bold scene headings, monospaced fonts, page breaks—everything a professional screenplay needs.

CUT TO:

INT. STUDIO - DAY

NARRATOR
(building excitement)
But here's where it gets really interesting. We didn't just build a PDF converter. We built an entire production pipeline tool.

Multiple terminal windows showing different export formats appear.

NARRATOR
Shot list generation—extracts every scene and shot heading into structured tables.

Shot list PDF with scene breakdown table appears on screen.

NARRATOR
Markdown, CSV, or PDF format. Your choice.

NARRATOR
Entity extraction—automatically catalogs every character, location, and prop mentioned in your script.

Entity inventory showing categorized characters and locations displays.

NARRATOR
Final Cut Pro integration—exports FCPXML timelines with scene markers and shot keywords.

Final Cut Pro timeline with imported markers appears.

NARRATOR
Giving you a head start on your edit before you've shot a single frame.

CUT TO:

INT. STUDIO - DAY

NARRATOR
(serious, principle-driven)
When we started this project, we had three core principles from the beginning:

ON-SCREEN TEXT

Text overlays appearing in sequence:

"Number one: Simplicity over complexity."

"Number two: Clarity over cleverness."

"Number three: Maintainability over optimization."

The CLAUDE.md file appears in the editor showing these principles.

NARRATOR
These weren't just nice-to-have guidelines—they were hard requirements documented in our codebase.

Code showing clean, simple function structures appears.

NARRATOR
And Claude Code helped us enforce them at every step.

CUT TO:

INT. DIGITAL WORKSPACE - DAY

NARRATOR
(conversational)
Here's how the collaboration worked. Instead of writing code from scratch, I would describe the feature I wanted—

ON-SCREEN

Chat conversation example appears showing a user request.

NARRATOR
"I need to parse scene headings from markdown that start with three hash marks."

NARRATOR
Claude Code would analyze the existing codebase, understand our architectural patterns, and suggest an implementation that fit our style.

SPLIT SCREEN

Claude Code on the left, code editor on the right, code being generated in real-time.

NARRATOR
But here's the key: it wasn't just generating code. It was having a conversation about design decisions.

ON-SCREEN
Key questions appear as text overlays:
"Should we use regex or a state machine for parsing?"
"How do we handle edge cases like nested parentheticals in dialogue?"
"What's the right abstraction for rendering engines?"
Code review session showing different approaches appears.

NARRATOR
(impressed)
This iterative design conversation is where the real magic happened. Not just code generation—actual architectural thinking.

CUT TO:

INT. STUDIO - DAY

NARRATOR
We required 80% code coverage for all new features.

Test coverage report showing 82% coverage displays.

NARRATOR
Claude Code didn't just write the implementation—it wrote the tests alongside it.

SPLIT SCREEN
Implementation code on left, test code on right, displayed side by side.

NARRATOR
When we added vector PDF rendering using ReportLab, Claude generated:

ON-SCREEN
Checklist appearing item by item:
- Unit tests for font loading
- Integration tests for the full pipeline
- Edge case handling for missing bold fonts

Pytest output showing all tests passing appears.

NARRATOR
Every commit followed the conventional commits specification.

Git commit history with feat:, fix:, refactor: prefixes displays.

NARRATOR
feat, fix, refactor. This meant our changelog was automatically generated and our version bumps were semantically correct.

CUT TO:

INT. DIGITAL WORKSPACE - DAY

NARRATOR
As we added features like user sessions and conversion history, the database schema evolved.

Database schema diagram animates, showing changes over time.

NARRATOR
But we never made manual schema changes. Every change went through proper migrations.

Migration files appear in file explorer.

NARRATOR
And we kept our database.dbml file up to date as the single source of truth.

ON-SCREEN
DBML file excerpt showing table definitions appears.

NARRATOR
Claude Code would update the schema documentation at the same time it generated migration scripts. No drift, no confusion, no "wait, what version is production running?"

CUT TO:

INT. STUDIO - DAY

NARRATOR
(technical but accessible)
One of the trickiest challenges was rendering. We needed two engines:

Code walkthrough of rendering modules appears.

NARRATOR
Vector rendering—using ReportLab for crisp, selectable text that looks razor-sharp at any zoom level.

CLOSE ON
Vector PDF zoomed in showing perfect text clarity.

NARRATOR
Raster rendering—using Pillow as a fallback with zero external dependencies beyond the Python standard library.

Raster PDF at same zoom level appears.

NARRATOR
The vector output is objectively better—smaller file size, searchable text, perfect fonts.

ON-SCREEN
Comparison stats showing file sizes and quality metrics.

NARRATOR
But the raster version just works even on systems without ReportLab installed.

Terminal shows successful PDF generation without ReportLab.

CUT TO:

INT. STUDIO - CONTINUOUS

NARRATOR
(detail-oriented)
Here's a detail most people wouldn't notice: bold scene headings.

CLOSE ON
PDF showing bold scene headings in detail.

NARRATOR
The tool attempts to auto-detect a bold variant of your monospace font by pattern-matching filenames.

ON-SCREEN
Font detection logic appears:
DejaVuSansMono-Regular.ttf → DejaVuSansMono-Bold.ttf

NARRATOR
If it can't find one? It simulates bold by drawing the text twice with a tiny horizontal offset.

Before and after comparison showing simulated bold effect.

NARRATOR
(appreciative)
Clever, right? This is the kind of thoughtful detail that emerged from iterating with Claude Code. It suggested the fallback approach when we were stuck on font variants.

CUT TO:

INT. DIGITAL WORKSPACE - DAY

NARRATOR
We didn't stop at the CLI. We built a FastAPI web application with drag-and-drop upload and a live markdown editor.

Web interface demo shows file being dragged, live preview updating.

NARRATOR
The entire stack runs in Docker containers deployed to a Raspberry Pi—yes, a Raspberry Pi—

Physical Raspberry Pi with network cables appears on screen.

NARRATOR
—using our local container registry.

NARRATOR
(security-focused)
Security features built in from day one:

ON-SCREEN
Security checklist appearing:
- File type validation (markdown only)
- 1MB upload limit
- Rate limiting (5 requests per minute)
- No file storage—everything processed in-memory

GitHub Actions workflow visualization appears.

NARRATOR
Our CI/CD pipeline runs tests on every commit, builds multi-platform Docker images for ARM and x86, and pushes to our registry—all automated through GitHub Actions.

CUT TO:

INT. STUDIO - DAY

NARRATOR
(reflective, wisdom-sharing)
Building ScriptMD2PDF with Claude Code taught us three major lessons about AI-assisted development:

Montage of development highlights: code, tests, deployments.

NARRATOR
Number one: AI is not a replacement for architectural thinking.

ON-SCREEN TEXT
"AI amplifies, doesn't replace"

NARRATOR
We still made the big decisions about rendering engines, data structures, and API design. But Claude Code accelerated implementation by 3-4x.

NARRATOR
Number two: Documentation and conventions multiply AI effectiveness.

CLAUDE.md, conventional commits, and DBML files appear.

NARRATOR
Our principles, conventional commits, and schema documentation meant Claude always knew what good code looked like for our project.

NARRATOR
Number three: Test-driven development works beautifully with AI assistance.

Test suite with 100+ tests passing displays.

NARRATOR
When you require tests for everything, the AI writes more robust code because it's thinking about edge cases from the start.

CUT TO:

INT. DIGITAL WORKSPACE - DAY

NARRATOR
(amazed)
Here's what still amazes me: we built a tool that handles screenplay parsing, vector PDF rendering, entity extraction, FCPXML timeline generation, web APIs, Docker deployment, and CI/CD automation...

Quick cuts showing each feature in action.

NARRATOR
...in a codebase that's clean, well-tested, and maintainable.

Code metrics display: test coverage, lines of code, complexity scores.

NARRATOR
(conviction)
That's the promise of AI-assisted development when you do it thoughtfully.

CUT TO:

Md2Script-Script

INT. STUDIO - DAY

NARRATOR
(inviting)
ScriptMD2PDF is free and open source on GitHub.

ON-SCREEN
GitHub URL appears: github.com/kevinmcaleer/scriptmd2pdf

NARRATOR
You can use the CLI tool locally, or try the web interface at md2script.kevs.wtf.

Quick demo of both CLI and web interface.

NARRATOR
If you're interested in AI-assisted development, building automation tools, or just want to see clean Python architecture in action—check out the repo.

ANIMATION
Subscribe button animation with bell icon appears.

NARRATOR
And if you found this valuable, subscribe for more deep dives into development tools, AI workflows, and building things that actually work in production.

CUT TO:

INT. STUDIO - DAY

NARRATOR
(inspirational)
The future of software development isn't AI replacing developers.

SPLIT SCREEN
Developer with Claude Code working together.

NARRATOR
It's AI amplifying what good developers can build.

FINAL DEMO
Markdown transforming to beautiful PDF on screen.

NARRATOR
We just proved it with ScriptMD2PDF.

NARRATOR
(direct)
Thanks for watching. Now go build something.

END SCREEN
Links to GitHub repo, web interface, and next video appear.

FADE OUT.: