

The Quiet Revolution in Document Rendering

How label-driven architectures are replacing template engines, one PDF at a time.

Hugo Palma · Senior Editor

March 2026

For decades, document generation has been trapped in the template paradigm. You write a template with placeholders, feed it data, and pray the output looks right. When it doesn't — and it rarely does on the first try — you fight with margins, font sizes, and page breaks until the result is tolerable.

The fundamental problem is coupling. Templates bind structure, style, and content into a single artifact. Change any one dimension and the other two resist. A new font breaks the layout. A longer paragraph overflows a box. A different paper size cascades through every element.

Labels as the Universal Joint

Label-driven rendering introduces an indirection layer that decouples content from presentation. Each piece of content carries a semantic label — not a style declaration. The rendering engine resolves labels to styles at render time using a theme, a flat map from label names to visual properties.

This means the same source document can produce radically different visual outputs simply by swapping the theme. A corporate report and a creative portfolio can share the same rendering core, the same operation types, and even the same source structure. Only the label-to-style map differs.

“The best architecture is the one where you can change your mind about the visual design without touching the data model.”

— *Software Design Principles, Third Edition*

The Operation Model

At the core sits a small set of primitive operations: text, row, bullet, divider, spacer, and block. Each operation carries its content and a label reference. The engine resolves the label against the active theme, applies the resulting style, and renders the primitive.

Pagination, cursor management, font registration, and style isolation happen automatically. The consumer never calculates a Y coordinate or checks whether content fits on the current page. The engine handles all of this through a simple contract: declare what you want, and the core figures out how to lay it out.

This is a fundamentally different model from template-based rendering. There is no page template to fill. There is no fixed grid to conform to. The document flows naturally from operation to operation, breaking pages when needed, respecting keep-together constraints, and maintaining typographic consistency throughout.

What This Means in Practice

A magazine article, an invoice, a resume, and a blockquote collection can all be rendered by the same engine. The difference between them is not in the code — it is in the source JSON and the theme. The engine does not know or care what kind of document it is rendering. It simply executes operations.

This separation makes the system composable. New document types require no new code in the rendering engine. They require only a new source JSON structure and a theme that defines the labels used by that structure. The engine remains stable while the ecosystem of document types grows around it.