ON DOCUMENT ENGINEERING

Why pagination is the hard part

And what it takes to get right in pure TypeScript

Every tool that makes a PDF has to answer the same question sooner or later: what happens when the content does not fit on one page. Most answer it badly. They draw until they run out of room, then start a fresh page wherever the cursor happened to land, splitting a heading from its first paragraph or slicing a table row clean in half. A document engine earns its name by answering that question well.

The last fifteen percent

Laying a single page out is the easy part. You measure each element, stack them, and stop. The trouble starts at the page boundary. A paragraph has to break between lines, never between a line and its own descenders. An image has to move as a whole rather than tear across the seam. A table has to carry its header onto the next page so the reader does not lose the column meanings. Each of these is simple in isolation and unforgiving in combination, and together they are the fifteen percent of the work that takes eighty-five percent of the time.

The naive approach mutates positions as it goes, nudging elements up and down until they happen to fit. It works until two elements disagree about who moves, and then it produces the subtle, maddening bugs that only surface on the third page of a long report. The durable approach computes the fragment of content that fits, hands back the remainder, and repeats. Constraints flow down once, sizes flow up once, and nothing is nudged after the fact.

> *A table that breaks mid-row, a heading stranded at the foot of a page: these are the details that separate a layout engine from a drawing API.*

Metrics, not guesses

Wrapping text correctly is impossible without knowing how wide each glyph is. Browsers know because they ship a font stack and a shaping engine. A library that wants to avoid a headless browser has to bring its own metrics. jasy parses the Adobe font-metric files for the standard fonts and reads the same tables straight out of any TrueType file you embed, so the width of a line is computed from the true advance of every character rather than estimated from an average.

This is also why the same text wraps identically whether it is being measured, drawn, or split across a

page. One line breaker feeds all three. When the measure and the draw disagree by even a fraction of a point, text that was sized to fit suddenly does not, and a word drops to a line of its own for no visible reason. Keeping a single source of truth is less a nicety than a requirement.

Flow as a first-class idea

The payoff for treating flow as a first-class idea, rather than a patch bolted onto a single-page layout, is that complex documents stop being special cases. An invoice whose line items run long simply continues onto a second page with its header intact. A report with a chapter of dense prose paginates without anyone thinking about it. The author describes the shape of the document, and the engine decides where the paper ends.

None of this requires a browser, a virtual machine, or a network call. It is a tree of components, a set of real font metrics, and a fragmentation pass that knows how to say: this much fits here, the rest goes next. That is the whole trick, and getting it right is the difference between a PDF you fight with and one you forget about.