

texwright: Fast, Adaptable Typesetting for Language-Model Agents

Thomas Garren

Sophia XT

October 8, 2026

Abstract

texwright turns Markdown, a substantial subset of LaTeX, or a JSON document tree into PDF and HTML, and it is built for language-model agents. An agent writes one file, runs a check that answers in milliseconds, reads diagnostics that name the line and suggest the fix, and renders a typeset PDF in a fraction of a second. Paper, margins, body size, theme and authorship are set in the document itself, so adapting a document never needs a template, a style file or a command-line flag. Underneath is a typesetting engine written from first principles on Node’s standard library: TeX’s box and glue model, Knuth–Plass line breaking with hyphenation, the mathematics layout of *The TeXbook*, a page builder with floats and footnotes, and a PDF encoder that subsets OpenType fonts. It installs nothing else: no TeX, no browser and no third-party packages. On a benchmark of documents that language models wrote when asked for a PDF, 75% and 60% of two held-out sets came out clean on the first try, and 95% and 80% without errors. The agent can also look at its pages, drawn as images by texwright’s own rasterizer. This paper describes the document model, the readers, numbering and citations, the engine and the way it settles page references, the agent interface, the benchmark and the security model. texwright typeset this paper from a Markdown source.

Keywords: language-model agents, typesetting, LaTeX, Markdown, PDF, line breaking, diagnostics

Contents

1	Introduction	2
2	Design goals	2
3	The document model	3
4	Parsing	4
5	Numbering, references and citations	4
6	The typesetting engine	5
7	Page references	7
8	The agent interface	8
9	Writing as models write	10
10	Security model	10
11	Robustness and performance	11
12	Limitations	12
13	Conclusion	12

1 Introduction

TeX and LaTeX still set the standard for technical documents [1, 2], and they are hard to automate. A working installation runs to gigabytes, and when a document fails the error log speaks in terms of macro expansion rather than the line the author wrote. Pandoc [3] converts between almost any pair of formats, but to make a PDF it hands the document to a LaTeX engine and inherits that engine’s size and its errors. KaTeX [4] typesets mathematics quickly in a browser, but only mathematics.

A growing share of documents is now written by programs. A language-model agent asked for a report, a paper or a handout produces Markdown or LaTeX easily. What slows it down is everything after that: a toolchain to install, a failure it cannot interpret, and layout changes that mean editing a template it cannot see. texwright is designed for that loop. Each step is one call, each problem comes back as data, and each layout decision lives in the document the agent already writes.

Its contributions are:

1. One document model shared by three input formats (Markdown, LaTeX, JSON), so a document may mix Markdown and raw LaTeX freely (Section 3, Section 4).
2. LaTeX numbering semantics without TeX, including per-row equation numbers and section-relative theorem counters (Section 5).
3. A typesetting engine written from first principles: TeX’s box and glue model, Knuth–Plass line breaking, the mathematics layout of *The TeXbook*, a page builder with floats and footnotes, and a PDF encoder with font subsetting (Section 6).
4. Exact page references from repeated layout passes, with a fixed-point argument for when to stop (Section 7).
5. An interface built for agents: a check that answers in milliseconds, diagnostics with stable codes and fixes in the shape of an edit, layout chosen from the document’s own front matter, page images an agent can look at, a JSON Schema and an MCP server (Section 8).
6. A benchmark of documents that models wrote for a PDF, with held-out sets, and the reader extensions it led to: the packages, document classes and Markdown habits models use without being asked (Section 9).
7. A security model for documents written by untrusted authors, tested by an adversarial review (Section 10), and a test suite that fuzzes every reader (Section 11).

2 Design goals

Built for agents

One file in, one call, structured results out. The command-line interface and the MCP server [5] answer in JSON, a JSON Schema describes the document tree, and a one-page guide teaches the syntax.

Quick

Checking a document takes milliseconds and typesetting it takes a fraction of a second, so an agent can check after every edit and render as often as it likes.

Adaptable from the document

Paper, margins, body size, theme, running heads and authors are front matter keys in pandoc’s spelling, or the LaTeX class and geometry options models already know. Nothing needs a template.

No TeX and no borrowed engines

Nothing in the pipeline invokes TeX or a browser, and the published package runs no code it does not contain.¹ Node's standard library is the only foundation.

LaTeX semantics where readers notice them

Numbering, labels, references, citations, line breaks and page breaks behave as a LaTeX user expects.

Diagnostics as an API

A problem is data: a code that never changes meaning, a severity, a file, line and column, the offending line and a hint. Nothing a document does should raise an exception.

Safe with untrusted input

A document cannot read files outside its folder, run scripts, reach the network, or exhaust time or memory.

3 The document model

Every reader produces the same tree, and every later stage reads and rewrites it (Figure 1). Nodes are plain JSON objects tagged by a field `t`: blocks such as `Heading`, `Para`, `MathBlock`, `Table`, `Figure`, `Theorem`, and inlines such as `Text`, `Math`, `Ref`, `Cite` and `Note`. Every node may carry a source position, which is how a problem found three stages later still points at the line the author wrote.

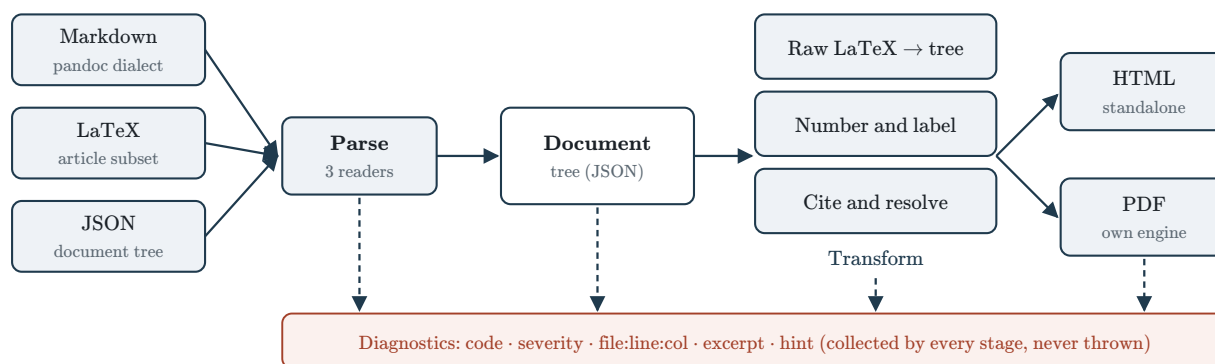


Figure 1: The pipeline. Three readers produce one document tree; transforms number, label and cite it; two writers render it. Every stage adds to one list of diagnostics instead of throwing.

Because the tree is plain JSON, a program can write a document directly in it. `texwright` schema prints the JSON Schema, and the reader validates input against it with errors that name the failing path:

```
{
  "texwright": 1,
  "meta": {
    "title": [
      {
        "t": "Text",
        "text": "Notes"
      }
    ]
  },
  "blocks": [
    {
      "t": "MathBlock",
      "tex": "e^{i\\pi}+1=0",
      "numbered": true,
      "label": "eq:euler",
    },
    {
      "t": "Para",
      "content": [
        {
          "t": "Text",
          "text": "By ",
        },
        {
          "t": "Ref",
          "label": "eq:euler",
          "style": "eq"
        }
      ]
    }
  ]
}
```

¹ Fonts and hyphenation patterns are data, not code: Latin Modern (GUST Font License), KaTeX's Computer Modern derivatives (SIL Open Font License) and TeX's US English patterns. Installed fonts fill in characters these lack.

4 Parsing

4.1 Markdown

The Markdown reader implements CommonMark [6] and GitHub’s tables, and adds what people use in practice from pandoc: YAML front matter, `$` and `$$` mathematics, attributes such as `{#fig:plot width=60%}`, fenced divisions for callouts and theorems, footnotes, definition lists, and citations such as `[@doe2020, p. 4]`. It was written from the specification: a block phase that matches each line against a stack of open containers, then an inline phase with a delimiter list for emphasis, both building texwright’s tree directly. A differential test against texwright’s earlier reader found identical trees for all 652 CommonMark and 713 GFM examples, every test input and 20,000 generated documents, and for 99.75% of 200,000 random ones; the rest are pathological inputs, such as tabs after nested quote markers, where the new reader follows the specification’s tab stops. The `$` rule follows pandoc: an opening `$` must be followed by a non-space and a closing `$` preceded by one and not followed by a digit, so “\$5 and \$10” stays text. Any LaTeX the reader does not handle itself is kept as a raw fragment for the LaTeX reader.

4.2 LaTeX

The LaTeX reader is a tokenizer and a recursive-descent parser written for the subset that articles use: sectioning, lists, figures and subfigures, `tabular` with `booktabs` rules, `\multicolumn` and `\multirow`, theorem environments declared with `\newtheorem`, `\input` and `\include`, the `natbib` and `biblatex` citation commands, and every display-math environment. Every token carries its source offset, so mathematics is cut from the source by position and never re-serialized. A stack of open groups and environments decides where each construct ends; when the source is broken, for example an unclosed `{` or an `\end` without its `\begin`, the stack reports the opening position and recovers, so later sections still parse.

User macros are expanded in text and recorded for mathematics. Expansion is bounded by depth, by count and by the total amount of text produced, which defeats definitions such as `\def\x{\x\x}`.

4.3 Mixing the two

Raw LaTeX inside a Markdown file is parsed by the LaTeX reader with a context whose macro table and metadata are the document’s own. A `\newcommand` in one fragment therefore works in the next, and `\bibliography{refs}` written in a Markdown file reaches the citation stage.

5 Numbering, references and citations

One pass in document order numbers sections (from the highest heading level the document uses, with letters after `\appendix`), figures and subfigures, tables, captioned listings, theorems and equations. Equations follow LaTeX exactly: `equation` takes one number, `align` and `gather` number every row unless the row says `\nonumber`, and `multline` numbers the block once. The rows below are numbered by this paper’s own pipeline:

$$u_t = \alpha u_{xx}, \tag{1}$$

$$\begin{aligned} u(0, t) &= u(1, t) = 0, \\ u(x, 0) &= f(x). \end{aligned} \tag{2}$$

The middle row carries no number, so (2) follows (1) directly. The mathematics engine keeps no counter of its own: the numbering pass decides the tag of every row and hands the tags to the engine with the formula, so the number printed and the number a reference resolves to come from the same place.

Theorem-like blocks share counters as `amsthm` declares them. With `\newtheorem{lemma}{theorem}[Lemma]`, lemmas count with theorems; with `\newtheorem{theorem}{Theorem}[section]`, both are numbered within sections, so a lemma and a theorem in the first section are 1.1 and 1.2.

A reference to a label that does not exist is an error (REF001) whose hint lists the closest existing labels by edit distance. A label defined twice is REF002; a label whose prefix contradicts its target, such as `fig:` on a table, is REF003.

Table 1: Reference and citation forms. The middle column is not typed by hand: `texwright` resolved each entry while producing this page.

Written as	Prints	Source of the text
<code>@fig:passes</code>	Figure 3	kind name and number
<code>\ref{fig:passes}</code>	3	number
<code>\eqref{eq:heat}</code>	(1)	equation number in parentheses
<code>\pageref{eq:heat}</code>	4	page builder (Section 7)
<code>[@knuth1981]</code>	[7]	citation processor

Citations are read from BibTeX [8] or CSL-JSON [9]. The BibTeX reader recovers from malformed entries, resolves `@string` macros, `crossref` and `biblatex` dates, and turns accents and protective braces into Unicode text. Two styles are built in: a numeric style that compresses runs (“[1–3]”) and an author–year style that disambiguates with letters (“2020a”). Unknown keys are errors whose hint names the nearest keys.

6 The typesetting engine

The PDF writer follows TeX’s own architecture (Figure 2). A typesetter walks the document tree and produces TeX’s raw material: boxes, glue that can stretch and shrink, penalties that price a break, and marks that record anchors, links, footnotes and floats. Paragraphs become lines, lines and other blocks become a vertical list, the page builder cuts that list into pages, and a shipping stage turns pages into PDF drawing operations.

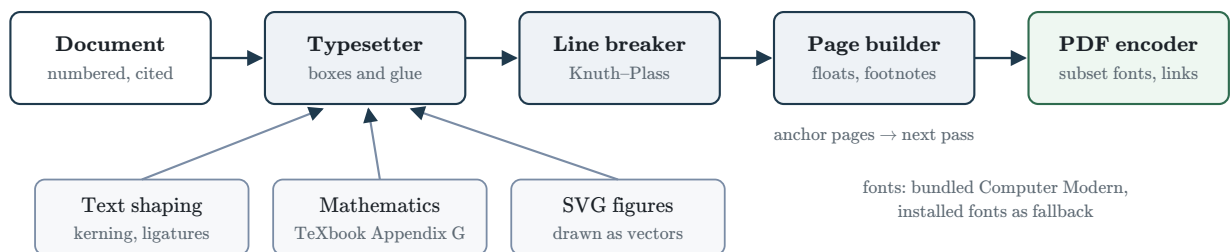


Figure 2: The typesetting engine. Text, mathematics and SVG figures all become boxes and glue; the line breaker and the page builder decide where lines and pages end; only the last stage knows about PDF.

6.1 Fonts and text

Text is set in Latin Modern, the OpenType form of LaTeX’s own Computer Modern, so it carries the ligatures, kerning and accented letters a LaTeX reader expects. A TrueType and OpenType reader handles `cmap` tables, metrics, `glyf` and CFF outlines, kerning from `kern` and GPOS, ligatures from GSUB, and font collections. Text is shaped against a stack of fonts: the face the style asks for, then the bundled Computer Modern faces, then installed fonts that cover the character. Where a face lacks an accented letter, the shaper builds it as TeX’s `\accent` does, and a character no available font has is reported (FONT001).

6.2 Paragraphs

Lines are broken with the Knuth–Plass algorithm [7], which chooses all the breaks of a paragraph together. For a line with adjustment ratio r the badness is $b = 100|r|^3$, and a break with penalty p on a line costs

$$d = \begin{cases} (l + b)^2 + p^2, & 0 \leq p < 10000, \\ (l + b)^2 - p^2, & -10000 < p < 0, \\ (l + b)^2, & p \leq -10000, \end{cases} \quad (3)$$

where l is the line penalty. As in TeX, a first pass tries without hyphenation, a second pass hyphenates by Liang’s method [10] with TeX’s own US English patterns, and a final pass adds emergency stretch so that a paragraph with an unbreakable word sets loosely rather than running into the margin. Interword spaces stretch more after a full stop, ligatures and kerns come from the font, and club and widow lines are priced with penalties.

6.3 Mathematics

The mathematics engine parses TeX notation into a tree of atoms and lays it out by the rules of Appendix G of *The TeXbook* [1]: atom classes and the spacing table between them, fraction and radical placement from the font’s parameters, sub- and superscript shifts, limits on large operators, and delimiters built to size from the extensible pieces in the bundled size fonts. The font parameters are the σ and ξ values of Appendix G for `cmsy10` and `cmex10`, scaled with the style. Alignment environments are boxes of columns, and an equation tag follows TeX’s `\eqno` rule: the formula is centred in the space the tag leaves, and a tag that does not fit beside its formula moves to a line of its own. Inline formulas join the paragraph, so a line can break after a relation or a binary operator at TeX’s penalties of 500 and 700. The same parser produces MathML for HTML output, so both writers accept exactly the same input. A formula that does not parse is reported (MATH001) at its line and column and printed as its source in red, so the rest of the document still comes out.

6.4 Pages

The page builder is TeX’s: it adds material to the current page until the page is full, prices each legal break by its badness and penalty, and cuts at the cheapest one. Headings carry a penalty that keeps them off the bottom of a page, list items and paragraphs are priced against widows and orphans, and a page that would overflow goes back to its best earlier break. Footnotes travel with the line that cites them and are set at the foot of that line’s page, split across pages only when they must be. Figures float: a figure that does not fit where it is cited goes to the top of the next page, and the text continues around it.

6.5 Figures

SVG figures are drawn as vectors by texwright’s own renderer. It parses the XML itself, refusing document type declarations and never expanding custom entities, then applies the CSS cascade (attribute selectors, structural pseudo-classes and sibling combinators included), transforms, use and `symbol`, markers, clip paths, masks and text. Arcs are converted to cubic Bézier curves, one per quarter turn. Gradients become PDF shading patterns; one that repeats or reflects is drawn over just the periods the shape spans, and stops that differ in opacity become a soft mask. An element with opacity is drawn as a transparency group unless it paints only once, when the opacity goes on that paint. Masks become soft masks and `mix-blend-mode` a group’s blend mode. Filters run on pixels: the preview rasterizer draws the element, the primitives run in linear RGB as SVG prescribes, and the result is embedded as an image with alpha. When a filter only adds something under the element, as a drop shadow does, only that part becomes an image and the element stays vector, its text still selectable. Generic serif text and the TeX family names use the bundled faces, so such labels match the document’s text, while a named family such as Georgia uses the installed font when there is one. Text can follow a path, and the HTML that diagram tools put in `foreignObject` is laid out as wrapped, aligned labels. JPEG images and most PNG images are embedded without recompressing their data; a PNG with transparency is decoded, and its alpha channel becomes a soft mask.

6.6 PDF output

The encoder writes objects and cross-reference tables directly [11]. Fonts are embedded as `Type0` fonts, TrueType with `CIDFontType2` descendants and OpenType with CFF outlines as `CIDFontType0`, both subset to the glyphs a document uses while keeping the original glyph identifiers, so the content streams address glyphs directly. The CFF subsetter replaces each unused charstring with a one-byte `endchar` and, after tracing which subroutines the kept glyphs call, each unused subroutine with a one-byte `return`, so no numbering changes; Latin Modern Roman shrinks from 112 KB to about 25 KB for a typical page. A `ToUnicode` map for each font keeps the text searchable and copyable, ligatures included [12]. Links, named destinations for every label, and bookmarks for every heading are written as the pages are shipped. Given a fixed date (the `SOURCE_DATE_EPOCH` convention), the same input produces the same bytes.

7 Page references

A table of contents near the start of a document lists pages that are only known once the whole document has been laid out, and printing those numbers can itself move text. texwright resolves this as LaTeX does with its auxiliary file, by repetition, but without writing anything to disk between passes (Figure 3).

The page builder records, for every anchor, the page and position where it landed. The first pass prints placeholders of a typical width in the contents and in every `\pageref`; each later pass prints the numbers the previous pass found. Only when two passes agree is the PDF written.

Proposition 1. *If two consecutive passes assign every anchor the same page, a further pass produces the same document.*

Proof. Typesetting is a deterministic function of the document tree and the page map from the previous pass. Equal maps give equal input, and therefore equal pages. $\square$

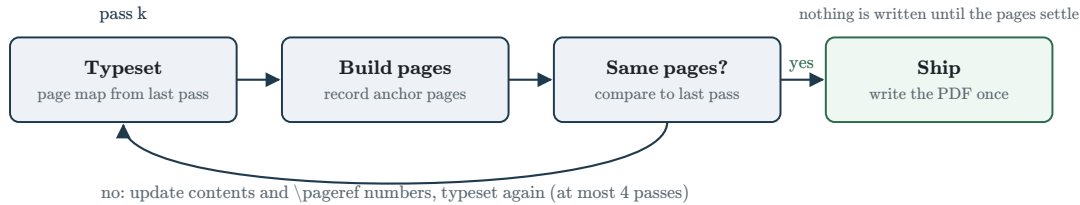


Figure 3: Page resolution. Each pass typesets the document with the page numbers found by the previous pass, the page builder records the page of every anchor, and the loop ends when nothing moves.

Termination in general is not guaranteed: a contents entry growing from page 9 to page 10 can widen a line and push a section to the next page. `texwright` stops after four passes and reports PDF002 if numbers are still moving. In practice two passes settle; this paper took two.

8 The agent interface

8.1 Diagnostics

Every stage appends to one list. A diagnostic records a stable code, a severity, the file, line and column, the source line, and where possible a hint. Codes are grouped by area (input, YAML, LaTeX, math, references, citations, images, fonts, layout, PDF, limits), and none changes meaning between releases. Repeated reports of one code are capped at 100 and summarized in a single line, so a document with forty thousand broken references produces a readable report in under a second.

```

{"ok": false, "errors": 1, "warnings": 0,
 "diagnostics": [{"code": "REF001", "severity": "error",
  "message": "Reference to undefined label \"sec:intr\"",
  "file": "paper.md", "line": 3, "col": 5,
  "excerpt": "See @sec:intr and $x^{2}$.",
  "hint": "Did you mean \"sec:intro\"?"}]}

```

The engine reports layout problems the same way. A line that sticks out into the margin is LAY001 with the paragraph’s position, a character that no available font contains is FONT001, and an SVG feature drawn approximately is SVG001.

`texwright check` runs every stage except page layout (it typesets every formula and resolves every image and reference) and takes about 15 ms for the showcase document once loaded. That makes the loop in [Figure 4](#) cheap enough to run after every edit.

Many diagnostics carry a fix: replace this text with that, at this line and column, which is exactly an edit. Fixes that cannot be wrong (a label one transposition away from the only close match, a misspelt command with one clear neighbour) are marked safe, and `texwright check --fix` applies them, repeating until none is left. `--diagnostics compact` puts each problem on one line to save tokens, and `texwright outline` lists headings, figures, tables and equations with their lines and reference counts, so an agent can find its way in a long document without reading all of it.

The same loop is available over MCP [5]. The server exposes `render_document`, `preview_page`, `check_document`, `fix_document`, `outline_document`, `new_document`, `get_schema` and `get_guide`; a test

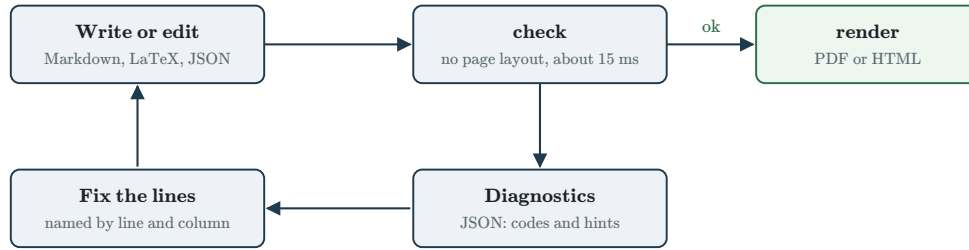


Figure 4: The repair loop an agent runs: edit, check, read the diagnostics, fix the named lines, and render when the check passes.

drives it with the official MCP client to check interoperability.

8.2 Looking at the result

Diagnostics cannot say that a table is too wide to read or that a figure landed two pages after its text. An agent that can see images can judge that itself, so `texwright` draws the pages it would write to the PDF as PNG images. The layout is the PDF’s own: the same passes produce the same pages, which are shipped to a rasterizer instead of the PDF writer. Glyphs are filled from their font outlines (TrueType quadratics and CFF charstrings), vector figures are stroked and filled with exact-area coverage, and PNG and baseline JPEG images are decoded and resampled, all in the same code base. On the showcase document and on this paper the images differ from `poppler`’s rendering of the PDF by 0.2 to 0.36 levels per channel on average, and a page takes about 20 ms to draw at 100 dpi. A PDF render also reports the page each heading, figure, table and numbered equation landed on, so an agent knows which page to look at.

8.3 Adapting a document

An agent adapts a document by editing the document. The keys below are `pandoc`’s, which models have seen many times; LaTeX sources get the same from `\documentclass[12pt,a4paper]{article}` and `\usepackage[margin=2cm]{geometry}`.

```

---
title: Quarterly results
author:
  - name: Thomas Garren
    affiliation: Sophia XT
papersize: a4
geometry: margin=2cm
fontsize: 12pt
theme: modern
---
```

A `fontsize` scales every size, skip and indent of the theme together, as LaTeX’s class options do. Options given on the command line or to the MCP tool win over the document, so a pipeline can still impose a house style. A value `texwright` cannot use is `OPT001` with a hint, like any other mistake, and `texwright check` reports it before anything is rendered.

9 Writing as models write

A tool built for agents should be measured on what agents write. We asked models, each with a one-line request such as “write me homework solutions for a linear algebra problem set, I’ll turn it into a PDF”, for 80 documents in four sets of 20: papers, homework, lecture notes, lab reports, CVs, letters, exams, handouts, design documents, recipes, policy briefs. They were told to write for pandoc or an ordinary LaTeX installation and not to run any converter, and they supplied the bibliographies and SVG figures their documents cite. The runner in the repository’s `bench/` folder renders each document unchanged and counts it clean when it has no errors and no warnings.

Sets A and C were the development sets: their failures said what to build. Set B was held out until A and C passed, then run once. Set D was written after that by a model told to report nothing about its contents, and run once at the end. [Table 2](#) gives the results.

Table 2: Documents written by language models for a PDF, rendered unchanged.

Set	Role	Clean on the first try	No errors
A + C	development, first run	29 / 40	37 / 40
B	held out	15 / 20	19 / 20
D	held out	12 / 20	16 / 20
A to D	after the fixes they led to	79 / 80	79 / 80

Markdown fared far better than LaTeX: 9 of 10 Markdown documents in D were clean, against 3 of 10 in LaTeX. The failures were rarely mistakes. Models reach for packages without asking whether the reader has them: `\SI{9.81}{\meter\per\second\squared}`, `\mhchem` for chemistry (`\ce{2H2 + O2 -> 2H2O}`), the physics package for derivatives and bra-kets, `amsthm`’s `\qedhere`, `mathtools`’ paired delimiters. They use whole document classes (`exam` with answer keys, `letter`), `algorithmic` for pseudocode, `tcolorbox` for coloured boxes, `thebibliography` for references, and pandoc’s habits in Markdown (`header-includes`, a closing `# References` heading, `H~2~0`). `texwright` now reads all of these as their authors meant. Where a reading is a guess, such as `\R` taken as $\mathbb{R}$ when no definition exists, it says so with an info diagnostic that does not count against a clean result.

The counts miss what looks wrong without being an error, so every page of the first 60 documents was also looked at through the page images. That found a `siunitx` table column read as several columns, amounts of money such as “\$3.78 per visitor versus \$3.41” taken for a formula, `\paragraph` headings numbered and set apart instead of run in, `\hfill` ignored, and the slash of `\neq` drawn beside the equals sign instead of across it, a bug in the PDF writer’s text positioning that had affected every PDF. None of these raised a diagnostic.

The held-out numbers, 75% and 60% clean on the first try, are the ones to quote. Each set’s failures led to fixes, so a set written today would likely do better than D did; only another held-out set can show by how much. All 80 documents were written by models of one family, and other models’ habits may differ.

10 Security model

Documents are treated as untrusted, since an agent may have written them on someone else’s instructions. The defenses are deliberately simple:

- Files are read only from the document’s folder and folders the user adds with `--allow-path`. The check is made on the path’s text before any file system access, and again after symbolic

links are resolved. Images referenced from inside an SVG figure obey the same rule.

- Nothing is fetched from the network. A remote image is reported and left out, and raw HTML passes an allowlist before it reaches the HTML writer.
- Every expansion mechanism has a budget: TeX macros, mathematics macros (by count and by expanded size), footnote copies, BibTeX `@string` text, table column specifications, JSON depth, and SVG nodes, path commands and `use` references. The YAML reader refuses aliases outright.
- Font and image files are parsed defensively: a corrupt or hostile file produces a diagnostic, never a crash, and the parsers are fuzzed with random corruption of real files.
- The MCP server confines every path to a workspace root and runs each request in a worker thread with a time limit, so one hostile document can neither reach other files nor stall the server.

Before release an independent reviewer was asked to break texwright and to prove each finding with a reproduction. Table 3 lists what it found; every item is fixed and pinned by a regression test.

Table 3: Findings of the adversarial review.

Finding	Effect before the fix	Fix
Unclosed [in a <code>\newcommand</code> inside math <code>\input{../secret.txt}</code>	19-byte document hung the checker file outside the folder copied into output	scan position can only move forward folder confinement for every read
UNC image path <code>//host/share/x.png</code> YAML alias expansion	network access, credential leak on Windows gigabytes of memory from 0.5 KB	refused on its text, before any file access own YAML reader refuses aliases
Image named <code>100%.png</code> One macro used 1,900 times	uncaught exception, stack trace out of memory	decode failure becomes IMG001 refuse formulas expanding past 200,000 characters
Hostile Markdown patterns	minutes of parsing for tens of KB	linear scans with precomputed indexes
MCP <code>output_path</code> and <code>base_dir</code>	read and write anywhere	workspace root (PATH001)

11 Robustness and performance

The suite has over 1,500 tests. Beyond unit and end-to-end tests it fuzzes all three readers with deterministic random soups of syntax fragments (nothing may throw), feeds them pathological structures (twenty thousand nested braces, deep JSON, unclosed environments by the thousand), corrupts real font and image files at random, and pins the running time of every hostile pattern found during review. The engine’s parts are checked against independent implementations: font tables and CFF subsets against fontTools, the PDF output by reading it back with two unrelated PDF readers, and SVG figures against a browser. On 30 test figures covering gradients, masks, filters, text and the output of diagram tools, texwright’s preview differs from Chrome’s rendering by 0.7 levels in 255 per pixel on average, and the PDF itself, drawn by poppler, by 0.9; most of what remains is the anti-aliasing of text. The suite passes on Windows and on Linux, including a Linux system with no fonts installed at all, where every face comes from the package.

Table 4: Median times on a desktop PC with Node 22, after the program has loaded. `scripts/measure.mjs` in the repository reproduces them.

Task	Time
check the showcase document	15 ms
check the LaTeX article example	12 ms
showcase to standalone HTML	12 ms
showcase to PDF (3 pages)	62 ms
LaTeX article to PDF (2 pages)	36 ms
this paper to PDF (13 pages)	209 ms
check 2,000 paragraphs and 200 equations (132 KB)	95 ms
the same document to PDF (67 pages)	0.8 s
check 1 MB of mixed Markdown	0.9 s

Every PDF time in Table 4 includes all layout passes. The browser-based writer that this engine replaced needed 2.7 s for the showcase, most of it spent printing the document three times; the engine does the same job more than forty times faster, because a layout pass is a function call rather than a print.

12 Limitations

Hyphenation patterns are bundled for English only; other languages are set without hyphenation, which shows in narrow columns. The mathematics engine covers `amsmath` and `amssymb` but not `\cancel`, `\sideset` or `\multicolumn` inside arrays, and glyph metrics are measured from the outlines rather than read from TeX’s font metric files, so spacing can differ from TeX’s by a fraction of a point. Complex scripts that need shaping beyond ligatures and kerning (Arabic, the Indic scripts) and bidirectional text are not supported. The SVG renderer leaves out lighting, turbulence and displacement filters and lays out HTML labels without general CSS layout, and PDF files cannot yet be used as figures (IMG003): such figures should be exported as SVG or PNG. TikZ, pgfplots and other drawing packages are reported (LTX004) rather than rendered, since they would need a TeX engine; tikz-cd diagrams are drawn approximately, as a grid with straight arrows. Colour emoji are drawn in one colour. The citation styles are two; CSL style files are not interpreted.

13 Conclusion

texwright shows that what makes a LaTeX document correct (numbering, references, citations, line and page breaks, page-accurate contents) can be had without TeX and without a browser, from a code base small enough to read, and that a typesetting tool built for language-model agents can be quick enough to run after every edit and simple enough to adapt from the document alone. Equation (1) is on page 4; every number in this sentence and in the paper was computed by texwright, and none appears in its source.

References

- [1] D. E. Knuth, *The TeXbook*. Reading, MA: Addison-Wesley, 1984.
- [2] L. Lamport, *LaTeX: A Document Preparation System*, 2nd ed. Reading, MA: Addison-Wesley, 1994.
- [3] J. MacFarlane, “Pandoc: a universal document converter,” 2024. [Online]. Available:

<https://pandoc.org>

- [4] The KaTeX Authors, “KaTeX: fast math typesetting for the web,” 2024. [Online]. Available: <https://katex.org>
- [5] Anthropic, “Model Context Protocol Specification, revision 2025-06-18,” 2025. [Online]. Available: <https://modelcontextprotocol.io/specification/2025-06-18>
- [6] J. MacFarlane, “CommonMark Spec, version 0.31.2,” 2024. [Online]. Available: <https://spec.commonmark.org/0.31.2/>
- [7] D. E. Knuth and M. F. Plass, “Breaking paragraphs into lines,” *Software: Practice and Experience*, vol. 11, no. 11, pp. 1119–1184, 1981, doi: [10.1002/spe.4380111102](https://doi.org/10.1002/spe.4380111102).
- [8] O. Patashnik, “BibTeXing,” 1988, Documentation distributed with BibTeX.
- [9] Citation Style Language Project, “Citation Style Language, version 1.0.2,” 2022. [Online]. Available: <https://docs.citationstyles.org/en/stable/specification.html>
- [10] F. M. Liang, “Word Hy-phen-a-tion by Com-put-er,” PhD thesis, Stanford University, 1983.
- [11] International Organization for Standardization, “Document management – Portable document format – Part 1: PDF 1.7,” ISO, Manual ISO 32000-1:2008, 2008.
- [12] Microsoft Corporation, “OpenType Specification, version 1.9,” 2022. [Online]. Available: <https://learn.microsoft.com/typography/opentype/spec/>