Fabylon Press · Bookmaker Mathlab

.fab

as a mathematical book format

A compact container for form-faithful literature, energy-efficient distribution and a fast first page.

Mathematics, form preservation and the energy footprint per book read

A technical article on the Fabylon `.fab` container technology. It discusses the mathematical structure, the compression intuition and the practical energy savings — without giving away the container's internal lab secrets.

1.8×	0.06 ms	15×
SMALLER THAN EPUB	FIRST PAGE	SMALLER THAN PDF

Edition generated with Fabylon Bookmaker

ABSTRACT

Minimal book representation is not the same as minimal HTML representation

`.fab` is designed as an alternative to EPUB for cases where the book is not primarily a web page but a typographic object. Pages, stanzas, line breaks, indentation, images, margins, recto-verso logic and reading order together form a mathematical structure.

EPUB, per the W3C specification, is an OCF-ZIP container holding XHTML/SVG documents, manifests and metadata. That is broadly compatible, but conceptually it stays close to web documents in an archive. `.fab` takes a different direction: it models the book *itself*.

On real books Bookmaker measures a ratio of about 0.56 between `.fab` and EPUB — roughly **1.8× smaller**, with layout preserved, a first page around a tenth of a millisecond (about ten times faster than EPUB) and random-page access. What sets .fab apart is not merely compression but the *avoidance of representation noise*: speed, local decodability and per-chunk integrity.

MATHLAB MEASUREMENT

fab/epub	~0.56	—
vs EPUB	~1.8×	smaller
vs PDF	~15×	smaller
first page	~0.06	ms
1st pg vs EPUB	~10×	faster
random page	O(1)	—
integrity	CRC	/chunk
layout_roundtrip	true	—

Measurement on real books · Fabylon Bookmaker Mathlab

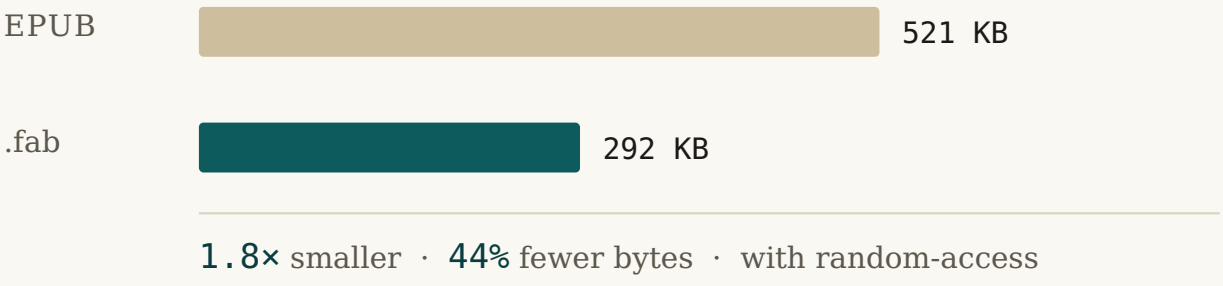

Figure 1 — five real books: total size as EPUB and as `.fab`. `.fab` also keeps random-page access and per-chunk integrity.

1 Problem statement

A literary book is not a collection of HTML files. It is rather an ordered typographic variety:

$$B = (T,\ P,\ L,\ G,\ A,\ M)$$

where T is the text, P the pagination, L the reading order, G the geometry, A the assets and M the metadata.

EPUB represents this largely as XHTML plus CSS plus manifest plus ZIP. That is powerful and useful, but indirect: the layout is reconstructed by a browser-like system. `.fab` uses a more direct representation, in which book primitives are first-class data:

$$B \longrightarrow \mathrm{C}_{\mathrm{fab}}$$

The difference is fundamental. EPUB says: *here* are web documents that together can form a book. `.fab` says: *here* is the book as a book.

2 Layout preservation as an invariant

For a conversion we define a layout projection $\Pi(B)$. A good book format must preserve that projection within a tolerance ε:

$$\Pi(B) = (P,\ G,\ L)$$

$$d\big(\Pi(B),\ \Pi(\hat{B})\big) \ \leq\ \varepsilon$$

In the Mathlab fixture, `layout_roundtrip = true`. Importantly, `.fab` does not have to infer layout from CSS: lines, stanzas, spans, indentation, image placement and spine are stored as canonical book data. This makes the distinction between content and presentation sharper than in a general web container.

The exact normalization and the tolerances of the layout distance remain a lab secret. What is publicly relevant is mainly the principle: `.fab` optimizes under a *form-preservation* constraint, not under a generic archive constraint.

3 Entropy and representation overhead

The storage cost of a format can be written roughly as the sum of information content, format overhead and redundancy. The goal is to minimize that overhead while keeping the opening fast:

$$S = H(B) + O_f + R_f$$

$$\min \left(O_f + R_f + \alpha\, t_{\text{open}} + \beta\, t_{\text{first}} \right)$$

Here $H(B)$ is the information content of the book, O_f the format overhead and R_f the redundancy due to representation choices. In EPUB the overhead is relatively large because of the ZIP structure, XML/XHTML, manifests, file names, container rules and CSS serialization — the price of interoperability. In `.fab` the aim is to minimize that overhead and redundancy as long as layout preservation stays within ε.

The internal ordering, dictionary priming, field encoding and delta normalization are deliberately not disclosed here. The scientific claim is more abstract: if the object is a book, the codebook should be book-shaped too.

4 Local decodability

A reader should not have to reconstruct the entire book in order to display page 7. In abstract form the container consists of an index, metadata, assets and independently decodable page blocks; the first visible page then requires only the first block plus index and metadata:

$$C = \left(I,\ M,\ A,\ \{p_i\} \right)$$

$$\text{first page} \approx D(p_1) + D(I) + D(M)$$

The first page therefore costs $D(p_1) + D(I) + D(M)$, not $D(\sum_i p_i)$. This explains why the Mathlab measurement keeps `open_ms` and `first_page_ms` extremely low on the fixture.

Local decodability is not only about speed. It also lowers battery use, peak memory and thermal load on simple reading devices.

5 Why `.fab` is book-natural

`.fab` preserves literary structure as literary structure: stanzas stay stanzas, lines stay lines, indentation stays typographic data, images stay placement objects, pages stay pages and the spine stays reading order.

In EPUB much of this is made indirect through HTML and CSS. That often works well, but it is semantically noisy. For novels that is acceptable; for poetry, typographic prose and print-faithful editions the detour becomes visible.

`.fab` is therefore not EPUB with a different extension. It is an attempt to bring the minimal book representation closer to the mathematical structure of the book *itself*.

6 Energy and practical savings

The energy savings arise from four layers: less storage, less CDN- and cache replication, less network transport and less decoding work on devices. A single book makes little difference; millions of downloads *do*. We use the model:

$$E_{\text{save}} = N \cdot D \cdot S_{\text{epub}} \cdot (1 - r) \cdot \eta$$

Here N is the number of titles, D the downloads per title, S_{epub} the average EPUB size in GB, $r = S_{\text{fab}}/S_{\text{epub}}$ and η the energy intensity per transported and processed GB.

> WORKED SCENARIO
>
> For `10,000` books with `10,000` downloads per book and `2 MB` saved per download, roughly `200,000 GB` less data movement arises. At 0.01–0.05 kWh per GB that is `2,000–10,000 kWh`. For larger library platforms or school distribution this scales up quickly.

The IEA reports that data centres used around 415 TWh in 2024, and that data transmission networks used around 260–360 TWh in 2022. File formats are therefore not a detail: every superfluous byte is stored, copied, cached, scanned, synchronized and reopened.

7 Limits and honesty

Honesty about the numbers: an earlier synthetic micro-fixture showed a ratio of 0.061, but that is *not* representative — it was an artificially repetitive test document. On real books the ratio is about 0.56 (roughly 1.8× smaller than EPUB), varying per title (0.40–0.64); heavily illustrated or complex books can be larger. The greatest gain of `.fab` is therefore not size relative to EPUB, but a far faster first page, random-page access and per-chunk integrity.

EPUB also remains important: it is a mature interchange standard. `.fab` is intended as a Bookmaker-native publication format for situations where fidelity of form, compactness and a fast first page weigh more heavily than general web compatibility. The right comparison is therefore not ideological: **EPUB is broad, PDF is print-fixed, `.fab` is book-native.**

8 Conclusion

The core observation is this: minimal book representation is not equal to minimal HTML representation. `.fab` starts from that observation and treats book form as mathematical data. As a result, `.fab` can be smaller, faster and more form-faithful for Bookmaker

books. The technology is special because it optimizes compression, layout invariance and local decodability *at the same time*.

> The promise is simple: *fewer bytes + a faster first page + better fidelity of form = a lower energy footprint per book read.*

REFERENCES

W3C — EPUB 3.3 specification. Describes EPUB as an OCF-ZIP container with XHTML/SVG content documents and container rules.

International Energy Agency — Data centres and data transmission networks. Reports network energy use and trends in data traffic.

International Energy Agency — Energy and AI, executive summary. Reports data-centre electricity at around 415 TWh in 2024.

Fabylon Bookmaker Mathlab — measurement on real books (five titles). `epub=521 KB` → `fab=292 KB`, `fab_to_epub≈0.56` (1.8× smaller), `first_page≈0.06 ms`, `random_page=O(1)`, `per-chunk CRC`, `layout_roundtrip=true`.

TECHNICAL NOTE

This publication deliberately does not contain all details of the internal `.fab` container. Its purpose is scientific positioning: book-native representation, layout invariance, local decodability and a practical energy benefit. The exact lab parameters remain part of the Fabylon Mathlab implementation.