Fabylon Press · Bookmaker Mathlab

.fab

als mathematisches Buchformat

Ein kompakter Container für formtreue Literatur, energieeffiziente Distribution und eine schnelle erste Seite.

Mathematik, Formerhalt und Energiepfad pro gelesenem Buch

Ein technischer Artikel über die Fabylon-`.fab`-Containertechnologie. Der Text behandelt die mathematische Struktur, die Kompressionsintuition und die praktische Energieeinsparung — ohne die internen Laborgeheimnisse des Containers preiszugeben.

1,8×	0,06 ms	15×
kleiner als EPUB	erste Seite	kleiner als PDF

Zusammenfassung

Minimale Buchrepräsentation ist nicht dasselbe wie minimale HTML-Repräsentation

`.fab` ist als Alternative zu EPUB konzipiert, wenn das Buch nicht in erster Linie eine Webseite, sondern ein typografisches Objekt ist. Seiten, Strophen, Zeilenfall, Einzüge, Bilder, Ränder, Recto-Verso-Logik und Lesereihenfolge bilden zusammen eine mathematische Struktur.

EPUB ist gemäß der W3C-Spezifikation ein OCF-ZIP-Container mit XHTML/SVG-Dokumenten, Manifesten und Metadaten. Das ist breit kompatibel, bleibt konzeptionell aber nah an Webdokumenten in einem Archiv. `.fab` schlägt eine andere Richtung ein: Es modelliert das Buch *selbst*.

Bei echten Büchern misst Bookmaker ein Verhältnis von etwa 0,56 zwischen `.fab` und EPUB — rund **1,8× kleiner**, bei erhaltenem Layout, einer ersten Seite um eine zehntel Millisekunde (etwa zehnmal schneller als EPUB) und Zugriff auf beliebige Seiten. Was .fab auszeichnet, ist nicht nur Kompression, sondern die *Vermeidung von Repräsentationsrauschen*: Geschwindigkeit, lokale Dekodierbarkeit und Integrität pro Chunk.

Mathlab-Messung

fab/epub	~0,56	—
vs EPUB	~1,8×	kleiner
vs PDF	~15×	kleiner
erste Seite	~0,06	ms
1. Seite vs EPUB	~10×	schneller
bel. Seite	O(1)	—
Integrität	CRC	/Chunk
layout_roundtrip	true	—

Messung an echten Büchern · Fabylon Bookmaker Mathlab

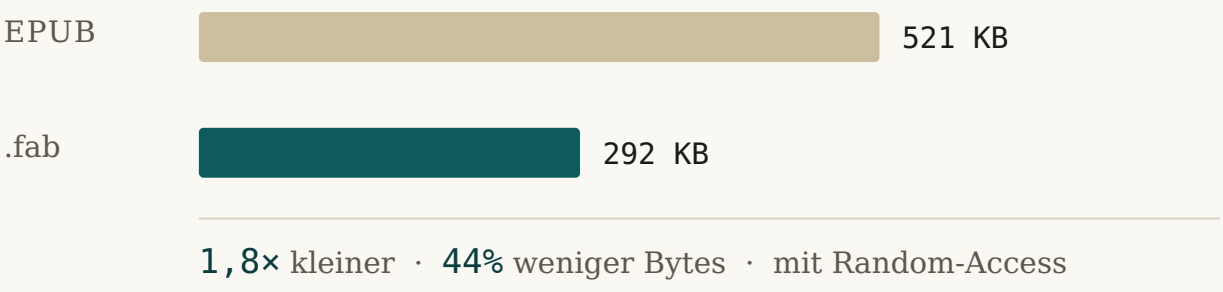

Abbildung 1 — fünf echte Bücher: Gesamtumfang als EPUB und als `.fab`. `.fab` behält zudem Zugriff auf beliebige Seiten und Integrität pro Chunk.

1 Problemstellung

Ein literarisches Buch ist keine Sammlung von HTML-Dateien. Es ist eher eine geordnete typografische Varietät:

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

wobei T der Text ist, P die Paginierung, L die Lesereihenfolge, G die Geometrie, A die Assets und M die Metadaten.

EPUB repräsentiert dies größtenteils als XHTML plus CSS plus Manifest plus ZIP. Das ist mächtig und nützlich, aber indirekt: Die Gestaltung wird von einem browserähnlichen System rekonstruiert. `.fab` verwendet eine direktere Repräsentation, in der Buchprimitive erstklassige Daten sind:

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

Der Unterschied ist grundlegend. EPUB sagt: *Hier* sind Webdokumente, die zusammen ein Buch bilden können. `.fab` sagt: *Hier* ist das Buch als Buch.

2 Layouterhalt als Invariante

Für eine Konversion definieren wir eine Layoutprojektion $\Pi(B)$. Ein gutes Buchformat muss diese Projektion innerhalb einer Toleranz ε bewahren:

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

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

In der Mathlab-Fixture gilt `layout_roundtrip = true`. Wichtig ist, dass `.fab` das Layout nicht aus CSS zurückrechnen muss: Zeilen, Strophen, Spans, Einzüge, Bildplatzierung und Spine werden als kanonische Buchdaten gespeichert. Dadurch wird die Unterscheidung zwischen Inhalt und Präsentation schärfer als bei einem allgemeinen Webcontainer.

Die genaue Normalisierung und die Toleranzen des Layoutabstands bleiben Laborgeheimnis. Öffentlich relevant ist vor allem das Prinzip: `.fab` optimiert unter einer *Formerhalt-Bedingung*, nicht unter einer generischen Archivbedingung.

3 Entropie und Repräsentations-Overhead

Die Speicherkosten eines Formats lassen sich grob als Summe aus Informationsgehalt, Format-Overhead und Redundanz schreiben. Ziel ist es, diesen Overhead zu minimieren und zugleich ein schnelles Öffnen zu erhalten:

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

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

Dabei ist $H(B)$ der Informationsgehalt des Buches, O_f der Format-Overhead und R_f die Redundanz durch die Repräsentationswahl. Bei EPUB ist der Overhead relativ groß durch ZIP-Struktur, XML/XHTML, Manifeste, Dateinamen, Containerregeln und CSS-Serialisierung — der Preis der Interoperabilität. Bei `.fab` wird versucht, diesen Overhead und die Redundanz zu minimieren, solange der Layouterhalt innerhalb von ε bleibt.

Die interne Ordnung, das Dictionary-Priming, die Feldkodierung und die Delta-Normalisierung werden hier bewusst nicht offengelegt. Die wissenschaftliche Behauptung ist abstrakter: Wenn das Objekt ein Buch ist, sollte auch das Codebuch buchförmig sein.

4 Lokale Dekodierbarkeit

Ein Reader sollte nicht das gesamte Buch rekonstruieren müssen, um Seite 7 anzuzeigen. In abstrakter Form besteht der Container aus einem Index, Metadaten, Assets und eigenständig dekodierbaren Seitenblöcken; die erste sichtbare Seite verlangt dann nur den ersten Block plus Index und Metadaten:

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

$$\text{erste Seite} \approx D(p_1) + D(I) + D(M)$$

Die erste Seite kostet also $D(p_1) + D(I) + D(M)$, nicht $D(\sum_i p_i)$. Das erklärt, warum die Mathlab-Messung `open_ms` und `first_page_ms` auf der Fixture extrem niedrig hält.

Lokale Dekodierbarkeit bedeutet nicht nur Geschwindigkeit. Sie senkt auch den Akkuverbrauch, die Speicherspitze und die thermische Last auf einfachen Lesegeräten.

5 Warum `.fab` buchnatürlich ist

`.fab` bewahrt literarische Struktur als literarische Struktur: Strophen bleiben Strophen, Zeilen bleiben Zeilen, Einzüge bleiben typografische Daten, Bilder bleiben Platzierungsobjekte, Seiten bleiben Seiten und der Spine bleibt Lesereihenfolge.

Bei EPUB wird vieles davon über HTML und CSS indirekt gemacht. Das funktioniert oft gut, ist aber semantisch verrauscht. Für Romane ist das akzeptabel; für Lyrik, typografische Prosa und drucktreue Ausgaben wird der Umweg sichtbar.

`.fab` ist daher kein EPUB mit anderer Endung. Es ist ein Versuch, die minimale Buchrepräsentation näher an die mathematische Struktur des Buches *selbst* zu bringen.

6 Energie und praktische Einsparung

Die Energieeinsparung entsteht aus vier Schichten: weniger Speicher, weniger CDN- und Cache-Replikation, weniger Netzwerktransport und weniger Dekodierarbeit auf Geräten. Ein einzelnes Buch macht wenig Unterschied; Millionen Downloads *schon*. Wir verwenden das Modell:

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

Dabei ist N die Anzahl der Titel, D die Downloads pro Titel, S_{epub} die durchschnittliche EPUB-Größe in GB, $r = S_{\text{fab}}/S_{\text{epub}}$ und η die Energieintensität pro transportiertem und verarbeitetem GB.

> RECHENSZENARIO
>
> Für `10.000` Bücher mit `10.000` Downloads pro Buch und `2 MB` Einsparung pro Download entstehen etwa `200.000 GB` weniger Datenbewegung. Bei 0,01–0,05 kWh pro GB sind das `2.000–10.000 kWh`. Bei größeren Bibliotheksplattformen oder Schuldistribution wächst dies schnell an.

Die IEA berichtet, dass Rechenzentren 2024 rund 415 TWh verbrauchten und dass Datenübertragungsnetze 2022 rund 260–360 TWh verbrauchten. Dateiformate sind also kein Detail: Jedes überflüssige Byte wird gespeichert, kopiert, gecacht, gescannt, synchronisiert und erneut geöffnet.

7 Grenzen und Ehrlichkeit

Ehrlichkeit zu den Zahlen: eine frühere synthetische Mikro-Fixture zeigte ein Verhältnis von 0,061, doch das ist *nicht* repräsentativ — es war ein künstlich stark wiederholendes Testdokument. Bei echten Büchern liegt das Verhältnis bei etwa 0,56 (rund 1,8× kleiner als EPUB), mit Streuung je Titel (0,40–0,64); stark illustrierte oder komplex gesetzte Bücher können höher liegen. Der größte Gewinn von `.fab` liegt daher nicht in der Größe gegenüber EPUB, sondern in einer viel schnelleren ersten Seite, Zugriff auf beliebige Seiten und Integrität pro Chunk.

Auch EPUB bleibt wichtig: Es ist ein ausgereifter Austauschstandard. `.fab` ist als Bookmaker-natives Publikationsformat für Situationen gedacht, in denen Formtreue, Kompaktheit und eine schnelle erste Seite schwerer wiegen als allgemeine Webkompatibilität. Der richtige Vergleich ist daher nicht ideologisch: **EPUB ist breit, PDF ist drucktreu, `.fab` ist buchnativ.**

8 Fazit

Die Kernbeobachtung lautet: Minimale Buchrepräsentation ist nicht gleich minimaler HTML-Repräsentation. `.fab` geht von dieser Beobachtung aus und behandelt die Buchform als mathematische Daten. Dadurch kann `.fab` kleiner, schneller und formtreuer für Bookmaker-Bücher sein. Die Technologie ist besonders, weil sie Kompression, Layoutinvarianz und lokale Dekodierbarkeit *zugleich* optimiert.

> Das Versprechen ist einfach: *weniger Bytes + eine schnellere erste Seite + bessere Formtreue = ein niedrigerer Energiepfad pro gelesenem Buch.*

Referenzen

W3C — EPUB-3.3-Spezifikation. Beschreibt EPUB als OCF-ZIP-Container mit XHTML/SVG-Inhaltsdokumenten und Containerregeln.

International Energy Agency — Data centres and data transmission networks. Berichtet über Netzwerk-Energieverbrauch und Trends im Datenverkehr.

International Energy Agency — Energy and AI, executive summary. Berichtet Rechenzentrumselektrizität von rund 415 TWh im Jahr 2024.

Fabylon Bookmaker Mathlab — Messung an echten Büchern (fünf Titel). `epub=521 KB → fab=292 KB`, `fab_to_epub≈0,56` (1,8× kleiner), `erste_Seite≈0,06 ms`, `beliebige_Seite=O(1)`, `CRC pro Chunk`, `layout_roundtrip=true`.

Technische Anmerkung

Diese Publikation enthält bewusst nicht alle Details des internen `.fab`-Containers. Ihr Zweck ist die wissenschaftliche Positionierung: buchnative Repräsentation, Layoutinvarianz, lokale Dekodierbarkeit und ein praktischer Energievorteil. Die genauen Laborparameter bleiben Teil der Fabylon-Mathlab-Implementierung.