Fabylon Press · Bookmaker Mathlab

.fab

als mathematisch boekformaat

Een compacte container voor vormvaste literatuur, energiezuinige distributie en een snelle eerste pagina.

Wiskunde, vormbehoud en energiepad per gelezen boek

Een technisch artikel over de Fabylon `.fab`-containertechnologie. De tekst bespreekt de mathematische structuur, de compressie-intuïtie en de praktische energiebesparing — zonder de interne labgeheimen van de container prijs te geven.

1,8×	0,06 ms	15×
kleiner dan EPUB	eerste pagina	kleiner dan PDF

Uitgave gegenereerd met Fabylon Bookmaker

SAMENVATTING

Minimale boekrepresentatie is niet hetzelfde als minimale HTML-representatie

`.fab` is ontworpen als alternatief voor EPUB wanneer het boek niet primair een webpagina is, maar een typografisch object. Pagina's, strofen, regelval, inspringingen, beelden, marges, recto-verso-logica en leesvolgorde vormen samen een mathematische structuur.

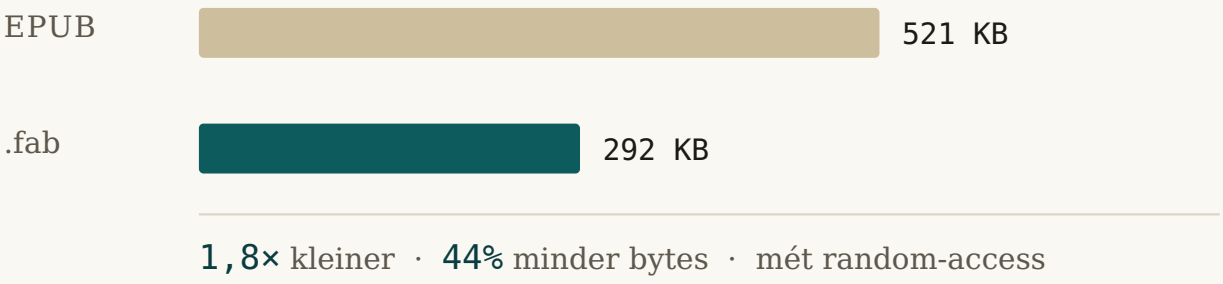
EPUB is volgens de W3C-specificatie een OCF-ZIP-container met XHTML/SVG-documenten, manifesten en metadata. Dat is breed compatibel, maar conceptueel blijft het dicht bij webdocumenten in een archief. `.fab` kiest een andere richting: het modelleert het boek zélf.

Op echte boeken meet Bookmaker een verhouding van ongeveer 0,56 tussen `.fab` en EPUB — circa **1,8× kleiner**, met behouden layout, een eerste pagina rond een tiende milliseconde (zo'n tien keer sneller dan EPUB) en willekeurige-pagina-toegang. Wat .fab onderscheidt is niet alleen compressie, maar het vermijden van representatieruis: snelheid, lokale decodeerbaarheid en per-chunk-integriteit.

MATHLAB-METING

fab/epub	~0,56	—
vs EPUB	~1,8×	kleiner
vs PDF	~15×	kleiner
eerste pagina	~0,06	ms
1e pag. vs EPUB	~10×	sneller
willek. pagina	O(1)	—
integriteit	CRC	/chunk
layout_roundtrip	true	—

Meting op echte boeken · Fabylon Bookmaker Mathlab



Figuur 1 — vijf echte boeken: totale omvang als EPUB en als `.fab`. `.fab` houdt bovendien willekeurige-pagina-toegang en per-chunk-integriteit.

1 Probleemstelling

Een literair boek is geen verzameling HTML-bestanden. Het is eerder een geordende typografische variëteit:

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

waarbij T de tekst is, P de paginering, L de leesvolgorde, G de geometrie, A de assets en M de metadata.

EPUB representeert dit grotendeels als XHTML plus CSS plus manifest plus ZIP. Dat is krachtig en nuttig, maar indirect: de vormgeving wordt gereconstrueerd door een browserachtig systeem. `.fab` gebruikt een directere representatie, waarin boekprimitieven eersteklas data zijn:

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

Het verschil is fundamenteel. EPUB zegt: híer zijn webdocumenten die samen een boek kunnen vormen. `.fab` zegt: híer is het boek als boek.

2 Layoutbehoud als invariant

Voor een conversie definiëren we een layoutprojectie $\Pi(B)$. Een goed boekformaat moet die projectie bewaren binnen een tolerantie ε:

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

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

In de Mathlab-fixture geldt `layout_roundtrip = true`. Belangrijk is dat `.fab` layout niet uit CSS hoeft terug te raden: regels, strofen, spans, inspringing, beeldplaatsing en spine worden bewaard als canonieke boekgegevens. Daardoor wordt het verschil tussen inhoud en presentatie scherper dan bij een algemene webcontainer.

De exacte normalisatie en de toleranties van de layoutafstand blijven labgeheim. Publiek relevant is vooral het principe: `.fab` optimaliseert onder een vórmbehoudconstraint, niet onder een generieke archiefconstraint.

3 Entropie en representatie-overhead

De opslagkosten van een formaat kunnen ruw worden geschreven als de som van informatie-inhoud, formaat-overhead en redundantie. Het doel is die overhead te minimaliseren onder behoud van een snelle opening:

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

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

Hier is $H(B)$ de informatie-inhoud van het boek, O_f de formaat-overhead en R_f de redundantie door representatiekeuze. Bij EPUB is de overhead relatief groot door ZIP-structuur, XML/XHTML, manifesten, bestandsnamen, containerregels en CSS-serialisatie — de prijs van interoperabiliteit. Bij `.fab` wordt geprobeerd die overhead en redundantie te minimaliseren zólang het layoutbehoud binnen ε blijft.

De interne ordening, dictionary-priming, veldcodering en delta-normalisatie worden hier bewust niet vrijgegeven. De wetenschappelijke claim is abstracter: als het object een boek is, moet het codeboek ook boëkvormig zijn.

4 Lokale decodeerbaarheid

Een reader moet niet het volledige boek reconstrueren om pagina 7 te tonen. In abstracte vorm bestaat de container uit een index, metadata, assets en zelfstandig decodeerbare pagina-blokken; de eerste zichtbare pagina vraagt dan alleen om het eerste blok plus index en metadata:

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

$$\text{eerste pagina} \approx D(p_1) + D(I) + D(M)$$

De eerste pagina kost dus $D(p_1) + D(I) + D(M)$, niet $D(\sum_i p_i)$. Dit verklaart waarom de Mathlab-meting `open_ms` en `first_page_ms` extreem laag houdt op de fixture.

Lokale decodeerbaarheid is niet alleen snelheid. Het verlaagt ook batterijgebruik, geheugenpiek en thermische belasting op eenvoudige leesapparaten.

5 Waarom `.fab` boeknatuurlijk is

`.fab` bewaart literaire structuur als literaire structuur: strofen blijven strofen, regels blijven regels, inspringing blijft typografische data, beelden blijven plaatsingsobjecten, pagina’s blijven pagina’s en de spine blijft leesvolgorde.

Bij EPUB wordt veel hiervan indirect gemaakt via HTML en CSS. Dat werkt vaak goed, maar het is semantisch luidruchtig. Voor romans is dat acceptabel; voor poëzie, typografisch proza en printgetrouwe uitgaven wordt de omweg zichtbaar.

`.fab` is daarom geen EPUB met een andere extensie. Het is een poging om de minimale boekrepresentatie dichter bij de wiskundige structuur van het boek zélf te brengen.

6 Energie en praktische besparing

De energiebesparing ontstaat uit vier lagen: minder opslag, minder CDN- en cache-replicatie, minder netwerktransport en minder decodeerwerk op apparaten. Een enkel boek maakt weinig verschil; miljoenen downloads wél. We gebruiken het model:

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

Hierbij is N het aantal titels, D de downloads per titel, S_{epub} de gemiddelde EPUB-grootte in GB, $r = S_{\text{fab}}/S_{\text{epub}}$ en η de energie-intensiteit per getransporteerde en verwerkte GB.

> Rekenscenario
>
> Voor `10.000` boeken met `10.000` downloads per boek en `2 MB` besparing per download ontstaat ongeveer `200.000 GB` minder databeweging. Bij 0,01–0,05 kWh per GB is dat `2.000–10.000 kWh`. Bij grotere bibliotheekplatforms of schooldistributie loopt dit snel op.

De IEA rapporteert dat datacenters in 2024 rond 415 TWh gebruikten, en dat datatransmissienetwerken in 2022 rond 260–360 TWh gebruikten. Bestandsformaten zijn dus geen detail: elke overbodige byte wordt opgeslagen, gekopieerd, gecachet, gescand, gesynchroniseerd en opnieuw geopend.

7 Grenzen en eerlijkheid

Eerlijkheid over de cijfers: een eerdere synthetische micro-fixture liet een ratio van 0,061 zien, maar dat is *niet* representatief — het was een kunstmatig sterk herhalend testdocument. Op echte boeken is de verhouding ongeveer 0,56 (circa 1,8× kleiner dan EPUB), met spreiding per titel (0,40–0,64); zwaar geïllustreerde of complex opgemaakte boeken kunnen hoger uitkomen. De grootste winst van `.fab` zit dan ook niet in grootte-ten-opzichte-van-EPUB, maar in een veel snellere eerste pagina, willekeurige-pagina-toegang en per-chunk-integriteit.

Ook blijft EPUB belangrijk: het is een volwassen uitwisselingsstandaard. `.fab` is bedoeld als Bookmaker-native publicatieformaat voor situaties waar vormtrouw, compactheid en een snelle eerste pagina zwaarder wegen dan algemene webcompatibiliteit. De juiste vergelijking is daarom niet ideologisch: **EPUB is breed, PDF is printvast, `.fab` is boeknative.**

8 Conclusie

De kernobservatie luidt: minimale boekrepresentatie is niet gelijk aan minimale HTML-representatie. `.fab` vertrekt vanuit die observatie en behandelt boekvorm als mathematische data. Daardoor kan `.fab` kleiner, sneller en vormgetrouwer zijn voor

Bookmaker-boeken. De technologie is bijzonder omdat zij compressie, layoutinvariantie en lokale decodeerbaarheid tégelijk optimaliseert.

> De belofte is eenvoudig: *minder bytes + snellere eerste pagina + betere vormtrouw = lager energiepad per gelezen boek.*

Referenties

W3C — EPUB 3.3 specification. Beschrijft EPUB als OCF-ZIP-container met XHTML/SVG-contentdocumenten en containerregels.

International Energy Agency — Data centres and data transmission networks. Rapporteert netwerkenergiegebruik en trends in dataverkeer.

International Energy Agency — Energy and AI, executive summary. Rapporteert datacenterelektriciteit rond 415 TWh in 2024.

Fabylon Bookmaker Mathlab — meting op echte boeken (vijf titels). `epub=521 KB` → `fab=292 KB`, `fab_to_epub≈0,56` (1,8× kleiner), `eerste_pagina≈0,06 ms`, `willekeurige_pagina=O(1)`, `per-chunk CRC`, `layout_roundtrip=true`.

Technische noot

Deze publicatie bevat bewust niet alle details van de interne `.fab`-container. Het doel is wetenschappelijke positionering: boeknative representatie, layoutinvariantie, lokale decodeerbaarheid en praktisch energievoordeel. De exacte labparameters blijven onderdeel van de Fabylon Mathlab-implementatie.