i 補足: この記事は Typst Advent Calender 2025 の 23 日目の記事です。

執筆環境

ソフト名	バージョン	補足
Typst	0.14	TYPST_FEATURES=html
Tinymist Typst	0.14.4	—

1. はじめに

Typst、とてもいいですよね。 Markdown のような手軽な書き心地でありながら、図表や脚注、参考文献まで美しく扱えるため気に入っています。

そんな Typst ですが、バージョン 0.14 で HTML エクスポート機能が大幅に強化されました。 セマンティックな要素のほとんど[1]が適切な HTML タグに変換されるようになったほか[1]、`html.elem` を使うことで任意の HTML タグを生成可能になりました。 つまり、Typst から HTML の DOM ツリーを直接操作できるようになったのです。

そこで今回はこの機能を活用して、Typst だけで記述・構築可能なブログシステムを作ってみました。 一般的な SSG（静的サイトジェネレータ）を使わず、記事の執筆からメタデータ管理までを Typst 内部で完結させる仕組みです。

なお本システムのスクリプトは WTFPL ライセンスで公開していますが、あくまで個人利用を目的とした実験的なものです。 その点はご了承ください。

2. ファイル構成

本ブログのファイル構成は以下の通りです（コード 1）。

ルートディレクトリには、システムの中核となる 5 つのファイルを配置しています。

- `index.typ`: トップページ（https://bibouroku.minimarimo3.jp）の生成元。
- `style.css`: サイト全体のデザイン定義。
- `posts.typ`: 全記事のメタデータ（タイトル、公開日、概要など）を集約管理するデータベース的なファイル。
- `template.typ`: 記事の共通レイアウトやデザインを定義したテンプレート。
- `build.py`: ビルドスクリプト。`posts.typ` から情報を読み取り、ディレクトリ走査や Typst コンパイル、静的ファイルの配置を一括で行います。

ビルドの成果物は `public` ディレクトリに出力され、この中身がそのまま Web サイトとして公開されます。

各記事は個別のディレクトリで管理しており、そこに執筆した `index.typ` を配置して `posts.typ` に登録することで、ビルド対象として認識される仕組みです。 なお、記事

[1] 例えばカスタム HTML 内での footnote など、一部未対応の機能もあります。

ファイル名を index.typ としているのは、ビルド後の index.html（ディレクトリへのアクセス時にデフォルトで表示されるファイル）に対応させるためです。

```
BIBOUROKU.MINIMARIMO3.JP
│  index.typ       # トップページ
│  style.css       # サイト全体のテーマを設定するファイル
│  posts.typ       # 公開対象の記事のメタデータを記述するファイル
│  template.typ    # 記事のテンプレート
│  build.py        # posts.typのデータをもとにディレクトリを走査し、ビルドと添付ファイルの移動を行うスクリプト
│
├─public           # ビルド後の出力先
│  │  index.html
│  │  style.css
│  │  feed.xml     # build.pyによって生成されます
│  │  sitemap.xml # build.pyによって生成されます
│  │
│  ├─Typstでブログを書く
│  │      index.html
│  │      index.pdf
│  │
│  └─テスト
│          index.html
│          index.pdf
│          テスト用画像.png
│
├─Typstでブログを書く # 記事1
│      index.pdf
│      index.typ
│      Typstでブログを書く.yaml
│
├─テスト              # 記事2
│      index.pdf
│      index.typ
│      reference.bib
│      テスト用画像.png
│
└─.github
  └─workflows
          deploy.yml
```

コード 1: 当ブログのファイル構成

3. 実装

3.1. **html.html** で出力される HTML の構造をカスタマイズ

Typst の HTML エクスポートは通常、文書内容を <body> タグ内に出力します。しかし、<head> 内に OGP タグや外部 CSS 読み込みを記述したい場合、これでは不十分です。 そこで template.typ では、html.html 関数を使用して <html> タグから始まる完全な DOM 構造を定義しました。

```
html.html(lang: "ja", {
  html.head({
    html.meta(charset: "utf-8")
    html.meta(name: "viewport", content: "width=device-width, initial-
scale=1")
    html.title(title)

    // OGP設定やGoogle Fontsの読み込み
    if description != "" {
      html.meta(name: "description", content: description)
    }
    html.elem("meta", attrs: (property: "og:title", content: title))
    html.link(rel: "preconnect", href: "https://fonts.googleapis.com")
    html.link(rel: "preconnect", href: "https://fonts.gstatic.com",
crossorigin: "anonymous")

    html.script(src: "/script.js")
    html.link(rel: "stylesheet", href: "/style.css")
  })

  html.body({
    html.div(class: "site-container", {
      // ヘッダー、記事本文、サイドバーなどを自由に配置
      html.main(class: "main-content", body)
      html.aside(class: "sidebar", { ... })
    })
  })
})
```

これにより、Typst だけで SEO 対策やスタイリングに必要な構造を自由自在に作り込むことができます。

3.2. その他の記事のランダム生成

ブログとしての回遊性を高めるため、記事下部にほかの記事を表示しています。 単にランダムに選ぶとビルドのたびに内容が変わってしまうため、記事タイトルのハッシュ値をシード（種）として使用し、乱数生成器を初期化することで、ランダムでありながら常に同じ結果が得られるように工夫しました。

```
// template.typより抜粋
import "@preview/suiji:0.5.0": *

// 自分以外かつ作成日が自分より若い記事が対象
let other-posts = post-data.pairs().filter(p => p.last().title !=
title).filter(p => p.last().create < create)

// 記事タイトルを数値化してシードにする
let seed = int(title.clusters().map(str.to-unicode).map(str).join().slice(0,
14))
let rng = gen-rng(seed)

// 記事リストをシャッフル
```

```
let (_, indices) = shuffle-f(rng, range(other-posts.len()))

// 上位3件を取得
let picks = indices.slice(0, 3).map(i => other-posts.at(i))
```

3.3. 記事の情報を Typst で管理する

トップページの記事一覧や RSS フィードを生成するには、全記事のメタデータ（タイトルや更新日）が必要です。 今回は `posts.typ` というファイルをデータベース代わりに使用するアーキテクチャを採用しました。

```
#let post-data = (
  "Typstでブログを書く": (
    title: "Typstでブログを書く",
    create: datetime(year: 2025, month: 12, day: 14),
    update: datetime(year: 2025, month: 12, day: 21),
    description: "Typst v0.14の新機能を使って、Typstだけでブログシステムを構築する試
み。",
    tags: ("Typst", "HTML"),
  ),
  "テスト": (
    title: "テスト",
    create: datetime(year: 2025, month: 12, day: 12),
    update: none,
    description: "サイトの表示テスト",
    tags: ("テスト",),
  ),
)

#metadata(post-data) <post-list>
```

Typst ファイル内からは `import` することで辞書としてデータを扱えます：

```
#import "../template.typ": project
#import "../posts.typ": post-data
#let meta = post-data.at("Typstでブログを書く")
#show: project.with(..meta)
```

一方、ビルドスクリプト（Python）からは `typst query` コマンドを使用することで同じ情報を JSON で取得できます。

```
result = subprocess.run(
    ["typst", "query", "posts.typ", "<post-list>"],
    capture_output=True,
    text=True,
    check=True,
    encoding="utf-8"
)
data = json.loads(result.stdout)
```

data はこんな感じ:

```
[
  {
```

```
    "func": "metadata",
    "value": {
      "Typstでブログを書く": {
        "title": "Typstでブログを書く",
        "create": "datetime(year: 2025, month: 12, day: 14)",
        "update": "datetime(year: 2025, month: 12, day: 21)",
        "description": "Typst v0.14の新機能を使って、Typstだけでブログシステムを構築する試み。",
        "tags": [
          "Typst",
          "HTML"
        ]
      },
      "テスト": {
        "title": "テスト",
        "create": "datetime(year: 2025, month: 12, day: 12)",
        "update": null,
        "description": "サイトの表示テスト",
        "tags": [
          "テスト"
        ]
      }
    },
    "label": "<post-list>"
  }
]
```

これにより、Markdown の Frontmatter のようなメタデータ管理を Typst の文法だけで統一して行えるようになりました。

3.4. 未実装機能への対処

HTML エクスポートは発展途上のため数式や脚注などで工夫が必要な場面がありました。

3.4.1. 数式(Math)を SVG 化して埋め込む

現状、数式を HTML に変換することはできないようです。 これについては、`html.frame` を使用して数式を一度フレーム（画像扱い）にし、SVG として HTML 内に埋め込む回避方法が Typst の issue に紹介されていたためこの方法を採用しています。

```
show math.equation.where(block: false): it => {
  html.elem("span", attrs: (role: "math"), html.frame(it))
}
show math.equation.where(block: true): it => {
  html.elem("figure", attrs: (role: "math"), html.frame(it))
}
```

3.4.2. カスタム HTML 構造での注釈(Footnote)

`html.html` で独自の DOM 構造を作ると、標準の `footnote` がエラーになる制限があります(コード 2)。 これに対しては、Typst の counter 機能を使って自前で脚注システムを実装することで解決しました。

```
   error: footnotes are not currently supported in combination with a custom
   `<html>` or `<body>` element
      ┌─ \\?\G:\マイドライブ\bibouroku.minimarimo3.jp\テスト\index.typ:97:16
      │
   97 │ これがノートを付けられる対象1#footnote[footnoteの中身1]
      │                                ^^^^^^^^^^^^^^^^^^^^^^^^
      │
      = hint: you can still use footnotes with a custom footnote show rule
```

コード 2: 独自の DOM 構造で footnote を使用した際に出るエラー

```
let note-counter = counter("my-footnote")
show footnote: it => {
  note-counter.step()
  let num = note-counter.get().first()
  // CSSでツールチップ表示するためのHTML構造を出力
  html.span(class: "footnote-wrapper", {
    html.span(class: "footnote-marker", "※" + str(num))
    html.span(class: "footnote-content", it.body)
  })
}
```

4. まとめ

Typst の HTML 生成機能はまだ実験的な側面もありますが、個人のブログやドキュメントサイト構築には十分実用できるレベルに達していると感じました。 何より、普段のドキュメント作成で慣れ親しんだ Typst 記法がそのまま Web サイトとして出力される体験は非常に快適です。

皆さんもぜひ、Typst で自分だけの Web サイトを作ってみてください！

参考文献

[1] L. Mädje と M. Haug, 「Typst: Typst 0.14: Now accessible – Typst Blog」, Typst. 参照: 2025 年 12 月 19 日. [Online]. 入手先: https://typst.app/blog/2025/typst-0.14#richer-html-export:~:text=Most%20semantic%20elements%20(those%20from%20the%20Model%20category)%20are%20now%20properly%20mapped%20to%20semantic%20HTML.%20We%27ve%20also%20improved%20handling%20of%20textual%20content%20in%20HTML%20export.