Vivliostyle CLI ドキュメント

Table of Contents

Vivliostyle CLI ドキュメント

Vivliostyle CLI ドキュメント

はじめに

はじめに

Vivliostyle CLI は、HTML やマークダウン文書を組版するためのコマンドラインインターフェイスです。1. https://docs.vivliostyle.org/#/ja/vivliostyle-viewerVivliostyle Viewer[*1] を内蔵し、出版物に適した高品質な PDF を生成します。

Vivliostyle プロジェクトを作成する

次のいずれかのコマンドを実行してください。質問される内容に答えると、プロジェクトが自動的に作成されます。

```
npm create book
yarn create book # For yarn users
pnpm create book # For pnpm users
```

> *[!NOTE] 事前に* 2. https://nodejs.org/*Node.js[*2]（v20以上）のインストールが必要です。Bun や Deno など Node.js 以外のランタイムを使用する場合は、それぞれのランタイムの使用手順にしたがってください。*

プロジェクトを作成すると、自動的に `vivliostyle.config.js` という名前の構成ファイルが作成されます。詳しい使用方法は 構成ファイル を参照してください。

コミュニティーテンプレート

Vivliostyle CLI はデフォルトで Minimal、Basic、Documentation、Novel、Academic、Magazine のテンプレートを用意しています。もし使いたい Vivliostyle Theme が独自のテンプレートを提供している場合、プロジェクト作成ステップで `Use templates from the community theme` を選ぶことでそのテンプレートからプロジェクトを作り始めることができます。

また、自分で用意したテンプレートを使用することもできます。`--template` オプションで指定したテンプレートをもとにプロジェクトを作成します。

```
npm create book -- --template gh:org/repo/templates/awesome-template
```

テンプレートの参照方法は 3. https://github.com/unjs/giget#readmegiget[*3] のドキュメントを参照してください。

構成ファイル

構成ファイル

複数の記事や章ごとのファイルをまとめて１つの出版物を構成するには、構成ファイルを利用します。`vivliostyle build` または `vivliostyle preview` コマンドを実行するとき、カレントディレクトリに構成ファイル `vivliostyle.config.js` があるとそれが使われます。また、`vivliostyle.config.json` というファイル名で 1. https://code.visualstudio.com/docs/languages/json#_json-with-commentsJSONC[*1]（コメント付き JSON）形式で構成ファイルを作成することもできます。

構成ファイルの作成

次のコマンドで構成ファイル `vivliostyle.config.js` を作成することができます。

```
vivliostyle init
```

これでカレントディレクトリに `vivliostyle.config.js` が生成されます。作成される `vivliostyle.config.js` ファイルは以下のようなものです。

```
// @ts-check
import { defineConfig } from '@vivliostyle/cli';

export default defineConfig({
  title: 'My Book Title',
  author: 'John Doe',
  language: 'en',
  image: 'ghcr.io/vivliostyle/cli:latest',
  entry: ['manuscript.md'],
});
```

構成ファイルの設定内容

構成ファイルの設定内容についてはファイル内のコメント（`//` ではじまる）に説明があります。それぞれの項目について主要な設定内容を紹介します。すべての設定内容については Config Reference を参照してください。

- **title**: 出版物のタイトル（例: `title: 'Principia'`）
- **author**: 著者名（例: `author: 'Isaac Newton'`）
- **language**: 言語（例: `language: 'en'`）。この指定があると HTML の `lang` 属性に反映されます。

Config Reference

Config Reference

The configuration files `vivliostyle.config.js` and `vivliostyle.config.json` accept the `VivliostyleConfigSchema` for configuring the Vivliostyle CLI. You can reference the configuration's type scheme from TypeScript files.

```
import { VivliostyleConfigSchema } from '@vivliostyle/cli';
```

Alternatively, you can use the `defineConfig` helper function to define the configuration.

```
import { defineConfig } from '@vivliostyle/cli';

export default defineConfig({ ... });
```

Config API

VivliostyleConfigSchema

Type definition

```
type VivliostyleConfigSchema =
  | BuildTask[]
  | BuildTask;
```

BuildTask

Properties

- `BuildTask`
  - `entry`: (string | ContentsEntryConfig | CoverEntryConfig | ArticleEntryConfig)[] | ArticleEntryConfig | string
    Entry file(s) of the document.
  - `title`: string
    Title of the document.

テーマと CSS

テーマと CSS

原稿に対してフォントや文字の大きさなどの装飾を加えるには、カスケーディングスタイルシート（CSS）を適用します（HTML ファイルと同様のやり方です）。

スタイルシートの追加の指定

HTML ファイルに指定されているスタイルシートに加えて、追加のスタイルシート（CSS ファイル）を使うには、`--style` オプションでスタイルシートを指定します。

```
vivliostyle build example.html --style additional-style.css
```

この方法で指定したスタイルシートは、HTML ファイルで指定されているスタイルシートと同様（1. https://developer.mozilla.org/ja/docs/Web/CSS/Cascade#%E4%BD%9C%E6%88%90%E8%80%85%E3%82%B9%E3%82%BF%E3%82%A4%E3%83%AB%E3%82%B7%E3%83%BC%E3%83%88作成者スタイルシート[*1]）の扱いで、よりあとに指定されたことになるので、CSS のカスケーディング規則により、HTML ファイルからのスタイルの指定を上書きすることになります。

ユーザースタイルシートの指定

2. https://developer.mozilla.org/ja/docs/Web/CSS/Cascade#%E3%83%A6%E3%83%BC%E3%82%B6%E3%83%BC%E3%82%B9%E3%82%BF%E3%82%A4%E3%83%AB%E3%82%B7%E3%83%BC%E3%83%88ユーザースタイルシート[*2]を使うには、`--user-style` オプションでスタイルシートを指定します。（ユーザースタイルシートは、スタイル指定に `!important` を付けないかぎり、制作者スタイルシートのスタイル指定を上書きしません。）

```
vivliostyle build example.html --user-style user-style.css
```

CSS の内容を直接指定

`--css` オプションを指定すると、追加したいスタイルシートを直接 CSS のテキストで渡すことができます。このオプションは、簡単なスタイルシートや CSS 変数を設定するのに便利です。

```
vivliostyle build example.html --css "body { background-color: lime; }"
```

表紙ページの作成

構成ファイル `vivliostyle.config.js` に `cover: 'image.png'` のような指定がある場合、表紙 HTML ファイル `cover.html` が生成されて、出版物の表紙ページとして追加されます。

生成される表紙 HTML ファイルの内容は次のようになります。

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Book Title</title>
    <style data-vv-style>
      body {
        margin: 0;
      }
      [role="doc-cover"] {
        display: block;
        width: 100vw;
        height: 100vh;
        object-fit: contain;
      }
      @page {
        margin: 0;
      }
    </style>
  </head>
  <body>
    <section role="region" aria-label="Cover">
      <img role="doc-cover" src="image.png" alt="Cover image" />
    </section>
  </body>
</html>
```

`cover` はオブジェクト形式でも指定でき、以下のようなプロパティが指定できます。

- `src` を指定すると、読み込ませる表紙画像を指定できます。
- `htmlPath` を指定すると、目次の HTML ファイルを `cover.html` 以外に出力します。また、`false` を設定することで表紙 HTML ファイルを出力させないこともできます。このようにした場合、PDFでの出力には表紙ページが含まれずに、EPUBやWebPub形式で出力した場合の表紙画像として設定されます。

目次の作成

目次の作成

- 1. https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/table-of-contents Example: table-of-contents[*1]

構成ファイル `vivliostyle.config.js` に `toc: true` の指定がある場合、目次 HTML ファイル `index.html` が生成されて、それが出版物の先頭のファイルになります。

生成される目次 HTML ファイルの内容は次のようになります。目次 HTML の `title` と `h1` 要素には、出版物のタイトル（構成ファイルの `title` で指定）が出力されます。

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Book Title</title>
    <link href="publication.json" rel="publication" type="application/ld+json" />
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <h1>Book Title</h1>
    <nav id="toc" role="doc-toc">
      <h2>Table of Contents</h2>
      <ol>
        <li><a href="prologue.html">Prologue</a></li>
        <li><a href="chapter1.html">Chapter 1</a></li>
        <li><a href="chapter2.html">Chapter 2</a></li>
        <li><a href="chapter3.html">Chapter 3</a></li>
        <li><a href="epilogue.html">Epilogue</a></li>
      </ol>
    </nav>
  </body>
</html>
```

`toc` はオブジェクト形式でも指定でき、以下のプロパティが指定できます。

- `htmlPath` を指定すると、目次の HTML ファイルを `index.html` 以外に出力します。
- `title` を指定すると、目次タイトル（`nav` 要素内の見出し `h2` 要素の内容 "Table of Contents"）を変更します。
- 目次には各エントリーへの参照以外にも、エントリー内の見出しも目次に含めることができます。そのようにしたい場合、`sectionDepth` に `1` から `6` の値（それぞれ `h1` から `h6` までのどのレベルを含めるか）を指定します。

特別な出力設定

特別な出力設定

EPUB 形式の出力

`vivliostyle build` コマンドに `-f`（`--format`）オプションで `epub` を指定する、または `-o`（`--output`）オプションで `.epub` 拡張子をつけて指定するとEPUBを出力します。

EPUB 形式で出力する場合、目次を設定した状態で出力することが推奨されます。目次の作成 を参考にして、構成ファイルに `toc` を設定した上で、以下のように出力オプションで EPUB を設定します。

```
vivliostyle build -o output.epub
```

また、次のように1回の `vivliostyle build` コマンドで PDF と EPUB の両方を生成することもできます（他の出力形式も同様です）。

```
vivliostyle build -o pdfbook.pdf -o epubbook.epub
```

Vivliostyle CLI では、EPUB 3 に準拠した EPUB ファイルを生成します。一方で、EPUB に適用する CSS ファイル自体はそのままの状態で出力されるため、EPUB ビューワーによっては表示に問題が発生する可能性があります。より多くの EPUB ビューワーに対応するためには、それぞれのビューワーに対応したテーマやCSSファイルを適用するようにしてください。日本語向けの EPUB の場合、1. https://dpfj.or.jp/counsel/guide電書協 EPUB3 制作ガイド[*1] に準拠した Vivliostyle Theme 2. https://github.com/vivliostyle/themes/tree/main/packages/%40vivliostyle/theme-epub3j"@vivliostyle/theme-epub3j"[*2] を用意しています。

Web 出版物（WebPub）の出力

`vivliostyle build` コマンドに `-f`（`--format`） オプションで `webpub` を指定すると、Web 出版物 (WebPub) を生成します。出力先 `-o`（`--output`）オプションには WebPub を配置するディレクトリを指定します。

```
vivliostyle build -o webpub/ -f webpub
```

生成された WebPub ディレクトリ内には出版物マニフェスト `publication.json` ファイルがあり、コンテンツの HTML ファイルの読み込み順などの情報が記述されています。これは W3C 標準仕様である 3. https://www.w3.org/TR/pub-manifest/Publication Manifest[*3] に準拠しています。

フロントエンドフレームワークのサポート

フロントエンドフレームワークのサポート

Vivliostyle CLIはWeb技術を活用して出版物を作成します。他のWebフロントエンドフレームワークと組み合わせることで、強力な機能を利用できます。たとえば、同じ原稿をWebページと出版物の両方にエクスポートできます。

静的にビルドされたHTMLファイルを参照する

フレームワークに静的サイトのビルド機能がある場合、ビルドされたHTMLファイルをVivliostyle CLIの`static`オプションで参照できます。これにより、そのファイルをエントリーとして読み込むことが可能です。

例として、`dist`ディレクトリにHTMLファイルがビルドされている場合を考えます。このディレクトリには`index.html`と、`blog`ディレクトリ内に`my-first-post.html`と`another-post.html`があります。これら3つのHTMLファイルから出版物を作成するには、以下のように設定します。

```
{
  static: {
    '/': 'dist',
  },
  entry: [
    '/index.html',
    '/blog/my-first-post.html',
    '/blog/another-post.html',
  ],
};
```

> *[!IMPORTANT]* `static` *でホスティングしたエントリーは、絶対パス（*`/`*ではじまるパス）で参照してください。そうしないと、*`static`*で指定したディレクトリではなく、*`vivliostyle.config.js`*からの相対パスでHTMLファイルを読み込もうとします。*

> *[!NOTE]* `entry` *に直接HTMLファイルを相対パスで指定する方法も動作しますが、*`static`*オプションを使うほうが便利です。たとえば、直接指定ではHTMLから参照される外部ファイル（画像など）が読み込めませんが、*`static`*オプションを使うとディレクトリ以下のファイルがすべて参照され、正しく表示されます。*

JavaScript API

JavaScript API

Exported members

Functions

- `build`
- `create`
- `createVitePlugin`
- `defineConfig`
- `preview`
- `VFM`

Interfaces

- `StringifyMarkdownOptions`
- `TemplateVariable`

Type Aliases

- `Metadata`
- `StructuredDocument`
- `StructuredDocumentSection`
- `VivliostyleConfigSchema`
- `VivliostylePackageMetadata`
- `VivliostylePackageMetadata`

Variables

- `readMetadata`

Functions

build()