Vivliostyle CLI ドキュメント

Table of Contents

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。Vivliostyle Viewer (https://docs.vivliostyle.org/ja/viewer/vivliostyle-viewer/)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 Node.js (https://nodejs.org/)（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 giget (https://github.com/unjs/giget#readme)のドキュメントを参照してください。

マニュアルインストール

次のコマンドでVivliostyle CLIをインストールできます。

```
npm install -g @vivliostyle/cli
```

上記のコマンドは、システム全体でVivliostyleを利用できるようにするものです。そうではなく、現在のディレクトリのみでVivliostyle CLIを利用する場合、以下のようにします。

```
npm install @vivliostyle/cli
```

PDFの生成

HTMLやMarkdownからPDFを生成

`vivliostyle build` コマンドでHTMLファイルを指定すると、HTMLから組版した結果のPDFファイルが出力されます。デフォルトで出力されるPDFファイル名は"output.pdf"です。

```
vivliostyle build index.html
```

同様に、Markdownファイルを指定するとMarkdownから組版した結果のPDFファイルが出力されます。

```
vivliostyle build manuscript.md -s A4 -o paper.pdf
```

Vivliostyle CLIで利用可能なMarkdown記法については、VFM: Vivliostyle Flavored Markdown (https://vivliostyle.github.io/vfm/#/)を参照してください。

出力PDFファイルの指定

`-o` (`--output`)オプションでPDFファイル名を指定できます。

```
vivliostyle build book.html -o book.pdf
```

WebのURLの指定

ローカルのHTMLファイルのほか、WebのURLを指定することもできます。

```
vivliostyle build https://vivliostyle.github.io/vivliostyle_doc/samples/gutenberg/Alice.ht
```

他の形式からPDFを生成

EPUBや 解凍されたEPUBのOPFファイル、`pub-manifest`（Web出版物のマニフェストJSONファイル）、`webbook`（目次やWeb出版物のマニフェストへのリンクがあるHTMLファイル）形式の読み込みに対応します。

```
vivliostyle build epub-sample.epub -o epub.pdf
vivliostyle build publication.json -o webpub.pdf
```

組版結果のプレビュー

`vivliostyle preview`コマンドで組版結果をブラウザでプレビューすることができます。プレビューを実行すると、ブラウザが立ち上がり組版結果をVivliostyle Viewerで閲覧することができます。

```
vivliostyle preview index.html
vivliostyle preview manuscript.md
vivliostyle preview epub-sample.epub
```

多数の文書から構成される出版物をすばやくプレビュー

多数の文書から構成される出版物をすばやくプレビューするためには、`-q`(`--quick`)オプションを指定してください。このオプションでは大まかなページ数カウントを使って迅速に文書をロードします（ページ番号の出力は不正確になります）。

```
vivliostyle preview index.html --quick
vivliostyle preview publication.json --quick
vivliostyle preview epub-sample.epub --quick
```

PDF形式以外の出力

Vivliostyle CLIはPDF形式以外にも、EPUB形式とWeb出版物（WebPub）の出力に対応します。詳細は特別な出力設定をご覧ください。

サポートする出力形式のマトリックスは以下の通りです。

入力\出力	`pdf`	`webpub`	`epub`
`pub-manifest`	🔵	🔵	🔵
`markdown`	🔵	🔵	🔵
`html` `webbook` (外部HTMLを含む)	🔵	🔵	🔵
`epub` `epub-opf`	🔵	🤦	🤦

その他のオプション

`vivliostyle help` コマンドでVivliostyle CLIで利用可能なオプションの一覧を表示できます。

```
vivliostyle help
vivliostyle help init
vivliostyle help build
vivliostyle help preview
```

秘密の機能: `vivliostyle` というコマンドの代わりに `vs` というコマンド名でも使用できるので、タイプ数を少し減らせます。

以下もご覧ください:

- Vivliostyle CLI (README) (https://github.com/vivliostyle/vivliostyle-cli/blob/main/README.md)

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 JSONC (https://code.visualstudio.com/docs/languages/json#_json-with-comments)（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。
- **size**:ページサイズ（例: `size: 'A4'`）。指定方法はページサイズの指定を参照してください。
- **theme**:ドキュメント全体に適用するVivliostyle Themesのパッケージ名、またはCSSファイルのパスを指定します。以下の値を指定することができ、複数のテーマを配列形式で指定することもできます。
  - npmスタイルのパッケージ名（例: `@vivliostyle/theme-techbook`, `./local-pkg`）
  - 単一のCSSを指定するURLやローカルファイルのパス（例: `./style.css`, `https://example.com/style.css`）
- **entry**:入力のMarkdownまたはHTMLファイルの配列を指定します。`js entry: [ { path: 'about.md', title: 'About This Book', theme: 'about.css' }, 'chapter1.md', 'chapter2.md', 'glossary.html' ],` entryには文字列またはオブジェクト形式で入力の指定ができます。オブジェクト形式の場合、以下のプロパティを追加できます。
  - `path`:エントリーのパスを指定します。このプロパティは必須ですが、`rel: 'contents'` または `rel: 'cover'` を指定するときのみ不要です。この指定については、以下の項目を参照してください
    - 目次を出版物の先頭以外の場所に出力するには
    - 表紙ページを出版物の先頭以外の場所に出力するには
  - `title`:エントリーのタイトルを指定します。このプロパティを設定しない場合、エントリーの内容からタイトルが取得されます。
  - `theme`:エントリーに適用するVivliostyle Themesのパッケージ名、またはCSSファイルのパスを指定します。
- **output**:出力先を指定。例: `output: 'output.pdf'`。デフォルトは `{title}.pdf`。次のように複数の出力を配列形式で指定することも可能です: `js output: [ './output.pdf', { path: './book', format: 'webpub', }, ],` outputには文字列またはオブジェクト形式で出力の指定ができます。オブジェクト形式の場合、以下のプロパティを追加できます。
  - `path`:出力先のパスを指定します。このプロパティは必須です。
  - `format`:出力するフォーマットを指定します（指定可能なオプション: `pdf`, `epub`, `webpub`）
    EPUB出力についてはEPUB形式の出力を、WebPub出力についてはWeb出版物（WebPub）の出力を参照してください。
- **workspaceDir**:中間ファイルを保存するディレクトリを指定。この指定がない場合のデフォルトはカレントディレクトリであり、Markdownから変換されたHTMLファイルはMarkdownファイルと同じ場所に保存されます。例: `workspaceDir: '.vivliostyle'`
- **toc**:このプロパティを指定すると、目次を含むHTMLファイル `index.html` が出力されます。詳しくは目次の作成を参照してください。

- **cover**:このプロパティを指定すると、表紙ページ用のHTMLファイル `cover.html` が出力されます。詳しくは表紙ページの作成を参照してください。
- **image**:使用するDockerのイメージを変更します。詳細はDockerを利用した生成を参照してください。

複数の入出力を一度に対応する

- Example: multiple-input-and-output (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/multiple-input-and-output)

以上の構成ファイルのオプションは、配列で複数指定することができます。配列として設定すると、複数の入力・出力を一度に扱うことができて便利です。 以下の例は、 `src` ディレクトリにあるMarkdownファイルから同名のPDFファイルに変換する構成ファイルです。

```
const fs = require('fs');
const path = require('path');

const inputDir = path.join(__dirname, 'src');
const outputDir = path.join(__dirname, 'output');
const files = fs.readdirSync(inputDir);

const vivliostyleConfig = files
  .filter((name) => name.endsWith('.md'))
  .map((name) => ({
    title: `Article ${path.basename(name, '.md')}`,
    entry: name,
    entryContext: inputDir,
    output: path.join(outputDir, `${path.basename(name, '.md')}.pdf`),
  }));
module.exports = vivliostyleConfig;
```

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.
  - `author`: string
    Author of the document.
  - `theme`: (ThemeConfig | string)[] | ThemeConfig | string
    Theme package path(s) or URL(s) of the CSS file.
  - `entryContext`: string
    Directory containing the referenced entry file(s).
  - `output`: (OutputConfig | string)[] | OutputConfig | string
    Output options.
  - `workspaceDir`: string
    Directory where intermediate files (e.g., manuscript HTMLs, publication.json) are saved. (default: `.vivliostyle`)
  - ~~`includeAssets`~~ *Deprecated*
    Use `copyAsset.includes` instead.
  - `copyAsset`: CopyAssetConfig
    Options for asset files to be copied when exporting output.
  - `size`: string
    PDF output size. (default: `letter`)
    - Preset: `A5`, `A4`, `A3`, `B5`, `B4`, `JIS-B5`, `JIS-B4`, `letter`, `legal`, `ledger`
    - Custom (comma-separated): `182mm,257mm` or `8.5in,11in`
  - ~~`pressReady`~~ *Deprecated*
    Use `pdfPostprocess.preflight: "press-ready"` instead
  - `pdfPostprocess`: PdfPostprocessConfig
    PDF post-processing options. When both pdfPostprocess and legacy options (pressReady, preflight, etc.) are specified, pdfPostprocess takes precedence.
  - `language`: string
    Language of the document.

- `readingProgression`: "ltr" | "rtl"
  Specifies the reading progression of the document. This is typically determined automatically by the CSS writing-mode. Use this option only if explicit configuration is needed.

- `toc`: TocConfig | boolean | string
  Options for Table of Contents (ToC) documents.

- ~~`tocTitle`~~ *Deprecated*
  Use `toc.title` instead

- `cover`: CoverConfig | string
  Options for cover images and cover page documents.

- `timeout`: number
  Timeout limit for waiting for the Vivliostyle process (in ms). (default: `300000`)

- `documentProcessor`: (option: import("@vivliostyle/vfm").StringifyMarkdownOptions, metadata: import("@vivliostyle/vfm").Metadata) => import("unified").Processor
  Custom function to provide a unified Processor for converting the source document to HTML.

- `documentMetadataReader`: (content: string) => import("@vivliostyle/vfm").Metadata
  Custom function to extract metadata from the source document content.

- `vfm`: VfmConfig
  Options for converting Markdown into a stringified format (HTML).

- `image`: string
  Docker image used for rendering.

- ~~`http`~~ *Deprecated*
  This option is enabled by default, and the file protocol is no longer supported.

- `viewer`: string
  URL of a custom viewer to display content instead of the default Vivliostyle CLI viewer. Useful for using a custom viewer with staging features (e.g., `https://vivliostyle.vercel.app/`).

- `viewerParam`: string
  Parameters for the Vivliostyle viewer (e.g., `allowScripts=false&pixelRatio=16`).

- `browser`: string
  Specify a browser type and version to launch the Vivliostyle viewer.

- `base`: string
  Base path of the served documents. (default: `/vivliostyle`)

- `server` : ServerConfig
  Options for the preview server.

- `static` : [key: (string)]: (string)[]
  Specifies static files to be served by the preview server. `js export default { static: { '/static': 'path/to/static', '/': ['root1', 'root2'], }, };`

- `temporaryFilePrefix` : string
  Prefix for temporary file names.

- `vite` : import("vite").UserConfig
  Configuration options for the Vite server.

- `viteConfigFile` : string | boolean
  Path to the Vite config file. If a falsy value is provided, Vivliostyle CLI ignores the existing Vite config file.

Type definition

```
type BuildTask = {
  entry:
    | (
        | string
        | ContentsEntryConfig
        | CoverEntryConfig
        | ArticleEntryConfig
      )[]
    | ArticleEntryConfig
    | string;
  title?: string;
  author?: string;
  theme?:
    | (ThemeConfig | string)[]
    | ThemeConfig
    | string;
  entryContext?: string;
  output?:
    | (OutputConfig | string)[]
    | OutputConfig
    | string;
  workspaceDir?: string;
  includeAssets?: string[] | string;
  copyAsset?: CopyAssetConfig;
  size?: string;
  pressReady?: boolean;
  pdfPostprocess?: PdfPostprocessConfig;
  language?: string;
  readingProgression?: "ltr" | "rtl";
  toc?: TocConfig | boolean | string;
  tocTitle?: string;
  cover?: CoverConfig | string;
  timeout?: number;
  documentProcessor?: (
    option: import("@vivliostyle/vfm").StringifyMarkdownOptions,
    metadata: import("@vivliostyle/vfm").Metadata,
  ) => import("unified").Processor;
  documentMetadataReader?: (
    content: string,
  ) => import("@vivliostyle/vfm").Metadata;
  vfm?: VfmConfig;
  image?: string;
  http?: boolean;
  viewer?: string;
  viewerParam?: string;
  browser?: string;
  base?: string;
  server?: ServerConfig;
  static?: {
    [key: string]: string[] | string;
```

```
  };
  temporaryFilePrefix?: string;
  vite?: import("vite").UserConfig;
  viteConfigFile?: string | boolean;
};
```

ContentsEntryConfig

Properties

- `ContentsEntryConfig`
  - `rel`: "contents"
  - `path`: string
  - `output`: string
  - `title`: string
  - `theme`: (ThemeConfig | string)[] | ThemeConfig | string
  - `pageBreakBefore`: "left" | "right" | "recto" | "verso"
    Specifies the page break position before this document. Useful for determining which side the first page of the document should be placed on in a two-page spread.
  - `pageCounterReset`: number
    Resets the starting page number of this document to the specified integer. Useful for controlling page numbers when including a page.

Type definition

```
type ContentsEntryConfig = {
  rel: "contents";
  path?: string;
  output?: string;
  title?: string;
  theme?:
    | (ThemeConfig | string)[]
    | ThemeConfig
    | string;
  pageBreakBefore?:
    | "left"
    | "right"
    | "recto"
    | "verso";
  pageCounterReset?: number;
};
```

ThemeConfig

Properties

- `ThemeConfig`
  - `specifier`: string
    The specifier name for importing the theme package or the path to a CSS file.
    - An npm-style package argument is allowed (e.g., `@vivliostyle/theme-academic@1`, `./local-pkg`).
    - A URL or a local path to a CSS file is allowed (e.g., `./style.css`, `https://example.com/style.css`).
  - `import`: (string)[] | string
    The path(s) to the CSS file(s) to import from the package. Specify this if you want to import files other than the default.

Type definition

```
type ThemeConfig = {
  specifier: string;
  import?: string[] | string;
};
```

CoverEntryConfig

Properties

- `CoverEntryConfig`
  - `rel`: "cover"
  - `path`: string
  - `output`: string
  - `title`: string
  - `theme`: (ThemeConfig | string)[] | ThemeConfig | string
  - `imageSrc`: string
  - `imageAlt`: string
  - `pageBreakBefore`: "left" | "right" | "recto" | "verso"
    Specifies the page break position before this document. Useful for determining which side the first page of the document should be placed on in a two-page spread.

Type definition

```
type CoverEntryConfig = {
  rel: "cover";
  path?: string;
  output?: string;
  title?: string;
  theme?:
    | (ThemeConfig | string)[]
    | ThemeConfig
    | string;
  imageSrc?: string;
  imageAlt?: string;
  pageBreakBefore?:
    | "left"
    | "right"
    | "recto"
    | "verso";
};
```

ArticleEntryConfig

Properties

- `ArticleEntryConfig`
  - `path`: string
  - `output`: string
  - `title`: string
  - `theme`: (ThemeConfig | string)[] | ThemeConfig | string
  - `encodingFormat`: string
  - `rel`: (string)[] | string
  - `documentProcessor`: (option: import("@vivliostyle/vfm").StringifyMarkdownOptions, metadata: import("@vivliostyle/vfm").Metadata) => import("unified").Processor
    Custom function to provide a unified Processor for converting the source document to HTML.
  - `documentMetadataReader`: (content: string) => import("@vivliostyle/vfm").Metadata
    Custom function to extract metadata from the source document content.

Type definition

```
type ArticleEntryConfig = {
  path: string;
  output?: string;
  title?: string;
  theme?:
    | (ThemeConfig | string)[]
    | ThemeConfig
    | string;
  encodingFormat?: string;
  rel?: string[] | string;
  documentProcessor?: (
    option: import("@vivliostyle/vfm").StringifyMarkdownOptions,
    metadata: import("@vivliostyle/vfm").Metadata,
  ) => import("unified").Processor;
  documentMetadataReader?: (
    content: string,
  ) => import("@vivliostyle/vfm").Metadata;
};
```

OutputConfig

Properties

- `OutputConfig`
  - `path`: string
    Specifies the output file name or directory. (default: `<title>.pdf`)
  - `format`: "pdf" | "epub" | "webpub"
    Specifies the output format.
  - `renderMode`: "local" | "docker"
    If set to `docker`, Vivliostyle will render the PDF using a Docker container. (default: `local`)
  - ~~`preflight`~~ *Deprecated*
    Use `pdfPostprocess.preflight` instead
  - ~~`preflightOption`~~ *Deprecated*
    Use `pdfPostprocess.preflightOption` instead
  - `pdfPostprocess`: PdfPostprocessConfig
    PDF post-processing options. When both pdfPostprocess and legacy options (pressReady, preflight, etc.) are specified, pdfPostprocess takes precedence.

Type definition

```
type OutputConfig = {
  path: string;
  format?: "pdf" | "epub" | "webpub";
  renderMode?: "local" | "docker";
  preflight?:
    | "press-ready"
    | "press-ready-local";
  preflightOption?: string[];
  pdfPostprocess?: PdfPostprocessConfig;
};
```

PdfPostprocessConfig

PDF post-processing options. When both pdfPostprocess and legacy options (pressReady, preflight, etc.) are specified, pdfPostprocess takes precedence.

Properties

- `PdfPostprocessConfig`
  - `preflight`: "press-ready" | "press-ready-local"
    Apply the process to generate a print-ready PDF.
  - `preflightOption`: (string)[]
    Options for the preflight process (e.g., `gray-scale`, `enforce-outline`). Refer to the press-ready documentation for more information: press-ready (https://github.com/vibranthq/press-ready)
  - `cmyk`: boolean | CmykConfig
    Convert device-cmyk() colors to CMYK in the output PDF. Can be a boolean or a config object with overrideMap and warnUnmapped options.
  - `replaceImage`: (ReplaceImageEntry)[]
    Replace images in the output PDF. Each entry specifies a source image path and its replacement image path. Useful for replacing RGB images with CMYK versions.

Type definition

```
type PdfPostprocessConfig = {
  preflight?:
    | "press-ready"
    | "press-ready-local";
  preflightOption?: string[];
  cmyk?: boolean | CmykConfig;
  replaceImage?: ReplaceImageEntry[];
};
```

CmykConfig

Properties

- `CmykConfig`
  - `overrideMap`: ("{tuple(Array)}")[]
    Custom RGB to CMYK color mapping. Each entry is a tuple of [rgb, {c, m, y, k}]. RGB can be an object {r, g, b} with integers (0-10000) or a hex color string (e.g. "#ff0000").
  - `reserveMap`: ("{tuple(Array)}")[]
    Pre-register RGB to CMYK color mappings for use in SVG or other non-CSS contexts. Each entry is a tuple of [rgb, {c, m, y, k}]. RGB can be an object {r, g, b} with integers (0-10000) or a hex color string (e.g. "#ff0000").
  - `warnUnmapped`: boolean
    Warn when RGB colors not mapped to CMYK are encountered. (default: true)
  - `mapOutput`: string
    Output the CMYK color map to a JSON file at the specified path.

Type definition

```
type CmykConfig = {
  overrideMap?: "{tuple(Array)}"[];
  reserveMap?: "{tuple(Array)}"[];
  warnUnmapped?: boolean;
  mapOutput?: string;
};
```

ReplaceImageEntry

Properties

- `ReplaceImageEntry`
  - `source`: string | RegExp
    Path to the source image file, or a RegExp pattern to match multiple files.
  - `replacement`: string
    Path to the replacement image file. When source is a RegExp, supports $1, $2, etc. for captured groups.

Type definition

```
type ReplaceImageEntry = {
  source: string | RegExp;
  replacement: string;
};
```

CopyAssetConfig

Properties

- `CopyAssetConfig`
  - `includes`: (string)[]
    Directories and files to include as asset files. Supports wildcard characters for glob patterns.
  - `excludes`: (string)[]
    Directories and files to exclude from asset files. Supports wildcard characters for glob patterns.
  - `includeFileExtensions`: (string)[]
    File extensions to include as asset files. (default: `[css, css.map, png, jpg, jpeg, svg, gif, webp, apng, ttf, otf, woff, woff2]`)
  - `excludeFileExtensions`: (string)[]
    File extensions to exclude from asset files.

Type definition

```
type CopyAssetConfig = {
  includes?: string[];
  excludes?: string[];
  includeFileExtensions?: string[];
  excludeFileExtensions?: string[];
};
```

TocConfig

Properties

- `TocConfig`
  - `title`: string
    Title of the generated ToC document.
  - `htmlPath`: string
    Location where the generated ToC document will be saved. (default: `index.html`)
  - `sectionDepth`: number
    Depth of sections to include in the ToC document. (default: `0`)
  - `transformDocumentList`: (nodeList: StructuredDocument[]) => (propsList: { children: any }[]) => any
    Function to transform the document list.
  - `transformSectionList`: (nodeList: StructuredDocumentSection[]) => (propsList: { children: any }[]) => any
    Function to transform the section list.

Type definition

```
type TocConfig = {
  title?: string;
  htmlPath?: string;
  sectionDepth?: number;
  transformDocumentList?: (
    nodeList: StructuredDocument[],
  ) => (
    propsList: { children: any }[],
  ) => any;
  transformSectionList?: (
    nodeList: StructuredDocumentSection[],
  ) => (
    propsList: { children: any }[],
  ) => any;
};
```

StructuredDocument

Properties

- `StructuredDocument`
  - `title` : string
  - `href` : string
  - `children` : (StructuredDocument)[]
  - `sections` : (StructuredDocumentSection)[]

Type definition

```
type StructuredDocument = {
  title: string;
  href: string;
  children: StructuredDocument[];
  sections?: StructuredDocumentSection[];
};
```

StructuredDocumentSection

Properties

- `StructuredDocumentSection`
  - `headingHtml` : string
  - `headingText` : string
  - `level` : number
  - `children` : (StructuredDocumentSection)[]
  - `href` : string
  - `id` : string

Type definition

```
type StructuredDocumentSection = {
  headingHtml: string;
  headingText: string;
  level: number;
  children: StructuredDocumentSection[];
  href?: string;
  id?: string;
};
```

CoverConfig

Properties

- `CoverConfig`
  - `src`: string
    Path to the cover image for the cover page.
  - `name`: string
    Alternative text for the cover image.
  - `htmlPath`: string | boolean
    Path where the generated cover document will be saved. (default: `cover.html`) If set to a falsy value, the cover document will not be generated.

Type definition

```
type CoverConfig = {
  src: string;
  name?: string;
  htmlPath?: string | boolean;
};
```

VfmConfig

Properties

- `VfmConfig`
  - `style`: (string)[] | string
    Path(s) or URL(s) to custom stylesheets.
  - `partial`: boolean
    Output markdown fragments instead of a full document.
  - `title`: string
    Title of the document (ignored in partial mode).
  - `language`: string
    Language of the document (ignored in partial mode).
  - `replace`: (test: RegExp; match: (result: RegExpMatchArray, h: any) => Object)[]
    string
    Handlers for replacing matched HTML strings.
  - `hardLineBreaks`: boolean
    Insert `<br>` tags at hard line breaks without requiring spaces.
  - `disableFormatHtml`: boolean
    Disable automatic HTML formatting.
  - `math`: boolean
    Enable support for math syntax.
  - `imgFigcaptionOrder`: "img-figcaption" | "figcaption-img"
    Order of img and figcaption elements in figure.
  - `assignIdToFigcaption`: boolean
    Assign ID to figcaption instead of img/code.
  - `footnote`: "pandoc" | "dpub" | "gcpm" | mode: "pandoc"
    "dpub" | "gcpm"
    Footnote output mode. Default is `'pandoc'` (endnote section).

Type definition

```
type VfmConfig = {
  style?: string[] | string;
  partial?: boolean;
  title?: string;
  language?: string;
  replace?: {
    test: RegExp;
    match: (
      result: RegExpMatchArray,
      h: any,
    ) => Object | string;
  }[];
  hardLineBreaks?: boolean;
  disableFormatHtml?: boolean;
  math?: boolean;
  imgFigcaptionOrder?:
    | "img-figcaption"
    | "figcaption-img";
  assignIdToFigcaption?: boolean;
  footnote?:
    | "pandoc"
    | "dpub"
    | "gcpm"
    | {
        mode:
          | "pandoc"
          | "dpub"
          | "gcpm";
      };
};
```

ServerConfig

Properties

- `ServerConfig`
  - `host`: boolean | string
    IP address the server should listen on. Set to `true` to listen on all addresses. (default: `true` if a PDF build with Docker render mode is required, otherwise `false`)
  - `port`: number
    Port the server should listen on. (default: `13000`)
  - `proxy`: [key: (string)]: import("vite").ProxyOptions[string]
    Custom proxy rules for the Vivliostyle preview server.
  - `allowedHosts`: (string)[] | boolean
    The hostnames that are allowed to respond to. Set to `true` to allow all hostnames. See `server.allowedHosts` option of Vite (https://vite.dev/config/server-options.html#server-allowedhosts) for more details.

Type definition

```
type ServerConfig = {
  host?: boolean | string;
  port?: number;
  proxy?: {
    [key: string]:
      | import("vite").ProxyOptions
      | string;
  };
  allowedHosts?: string[] | boolean;
};
```

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（作成者スタイルシート (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)）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

ユーザースタイルシート (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) 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; }"
```

ページサイズの指定

`-s`（`--size`)オプションでページサイズを指定できます。指定できるサイズは、A5, A4, A3, B5, B4, JIS-B5, JIS-B4, letter, legal, ledgerのいずれか、またはコンマで区切って幅と高さを指定します。

```
vivliostyle build paper.html -s A4 -o paper.pdf
vivliostyle build letter.html -s letter -o letter.pdf
vivliostyle build slide.html -s 10in,7.5in -o slide.pdf
```

このオプションは、`--css "@page { size: <size>; }"` と同等です。

トンボ（crop marks）の指定

`-m` (`--crop-marks`)オプションを指定すると、出力されるPDFにトンボ（印刷物の裁断位置を示す目印）が追加されます。

```
vivliostyle build example.html -m
```

`--bleed` オプションでトンボを追加したときの塗り足し幅を指定することができます。また、`--crop-offset` オプションで裁ち落とし線から外側の幅を指定することができます。

```
vivliostyle build example.html -m --bleed 5mm
vivliostyle build example.html -m --crop-offset 20mm
```

このオプションは、`--css "@page { marks: crop cross; bleed: <bleed>; crop-offset: <crop-offset>; }"` と同等です。

Vivliostyle Themesについて

- Vivliostyle Themes (https://vivliostyle.github.io/themes/)

Vivliostyle Themesは、Vivliostyleで出版物を作る際に使う公式のスタイルテーマ集です。Vivliostyle Themesを参照することで、自分でCSSを用意することなくスタイルを適用することができます。

テーマを見つける

npmパッケージとして公開されているテーマを見つけるには npm (https://www.npmjs.com/) でキーワード"vivliostyle-theme"を検索してください:

- List of Themes (npm) (https://www.npmjs.com/search?q=keywords%3Avivliostyle-theme)

テーマの利用

- Example: theme-css (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/theme-css)
- Example: theme-preset (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/theme-preset)

`-T`（`--theme`）オプション、または構成ファイルで `theme` を指定するとテーマを利用できます。ローカルにテーマファイルが存在しない場合、初回実行時に `themes` ディレクトリに自動的にインストールされます。

```
vivliostyle build manuscript.md --theme @vivliostyle/theme-techbook -o paper.pdf
```

また、ローカル環境にあるテーマを利用することもできます。単一のCSSファイルであれば、以下のように直接CSSファイルを指定します。

```
vivliostyle build manuscript.md --theme ./my-theme/style.css -o paper.pdf
```

また、ローカル環境にnpmに準拠した `package.json` ファイルがある場合、そのディレクトリにあるVivliostyle Themeを読み込むこともできます。以下は `my-theme` ディレクトリにVivliostyle Themeとして利用可能なパッケージが配置されているときの例です。

```
vivliostyle build manuscript.md --theme ./my-theme -o paper.pdf
```

Create Bookの利用

Create Bookを使用すると、あらかじめテーマが設定された状態のプロジェクトを簡単に作成できます。Create Book (https://docs.vivliostyle.org/ja/cli/getting-started/)を参照してください。

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。
- `name` を指定すると、カバー画像の代替テキストを変更します。

```
cover: {
  src: 'image.png',
  htmlPath: 'toc.html',
  name: 'My awesome cover image',
},
```

表紙ページを出版物の先頭以外の場所に出力するには

構成ファイル `vivliostyle.config.js` の `entry` の配列の要素として `{ rel: 'cover' }` を指定すると、その位置に表紙HTMLファイルが生成されます。

```
entry: [
  'titlepage.md',
  { rel: 'cover' },
  'chapter1.md',
  ...
],
cover: 'image.png',
```

これで、出版物の先頭の HTML ファイルは `titlepage.html` で、その次に目次の HTML ファイル `cover.html` という順番になります。

また、複数の表紙ページを追加することもできます。以下の例は、最初と最後のページにそれぞれ別の表紙ページを追加する例です。

```
entry: [
  {
    rel: 'cover',
    output: 'front-cover.html',
  },
  ...
  {
    rel: 'cover',
    output: 'back-cover.html',
    imageSrc: 'back.png',
  },
],
cover: 'front.png',
```

表紙ページをカスタマイズするには

- Example: customize-generated-content (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/customize-generated-content)

表紙ページをカスタマイズするには、次のように構成ファイルの `entry` の配列の要素として目次のファイルの `path` と `output`、`rel: 'contents'` を指定してください。

```
entry: [
  {
    path: 'cover-template.html',
    output: 'cover.html',
    rel: 'cover'
  },
  ...
],
```

そして、表紙のテンプレートとなるHTMLファイル `cover-template.html` を用意してください。`cover-template.html` の中に `<img role="doc-cover" />` というタグを用意することで、その部分に表紙画像が挿入された上で表紙のHTMLファイルが `cover.html` に出力されます。

目次の作成

- Example: table-of-contents (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/table-of-contents)

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。

```
toc: {
  htmlPath: 'toc.html',
  title: 'Contents',
  sectionDepth: 6,
},
```

目次を出版物の先頭以外の場所に出力するには

構成ファイル `vivliostyle.config.js` の `entry` の配列の要素として `{ rel: 'contents' }` を指定すると、その位置に目次HTMLファイルが生成されます。

```
entry: [
  'titlepage.md',
  { rel: 'contents' },
  'chapter1.md',
  ...
],
toc: 'toc.html',
```

これで、出版物の先頭の HTML ファイルは `titlepage.html` で、その次に目次の HTML ファイル `toc.html` という順番になります。

目次をカスタマイズするには

- Example: customize-generated-content (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/customize-generated-content)

目次をカスタマイズするには、次のように構成ファイルの `entry` の配列の要素として目次のファイルの `path` と `output`、`rel: 'contents'` を指定してください。

```
entry: [
  {
    path: 'toc-template.html',
    output: 'index.html',
    rel: 'contents'
  },
  ...
],
```

そして、目次のテンプレートとなる HTML ファイル `toc-template.html` を用意してください。`toc-template.html` の中に `<nav role="doc-toc"></nav>` というタグを用意することで、その部分に目次の項目が挿入された上で目次のHTMLファイルが `index.html` に出力されます。

詳しい目次の作り方については W3C Publication Manifest (https://www.w3.org/TR/pub-manifest/) 仕様に付属の Machine-Processable Table of Contents (https://www.w3.org/TR/pub-manifest/#app-toc-structure)を参照してください。

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、電書協EPUB3制作ガイド (https://dpfj.or.jp/counsel/guide)に準拠したVivliostyle Theme "@vivliostyle/theme-epub3j" (https://github.com/vivliostyle/themes/tree/main/packages/%40vivliostyle/theme-epub3j)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 Publication Manifest (https://www.w3.org/TR/pub-manifest/)に準拠しています。

WebPub は、Web 上で読むことができる出版物を作るのに使えます。また、次のように `publication.json` ファイルを `vivliostyle build` コマンドに指定することで、WebPubからPDFを生成することができます。

```
vivliostyle build webpub/publication.json -o pdfbook.pdf
```

印刷用PDF（PDF/X-1a形式）の生成

- Example: preflight (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/preflight)

`vivliostyle build` コマンドの `--preflight press-ready` オプション、または構成ファイルで `preflight: 'press-ready'` を指定すると印刷入稿に適したPDF/X-1a形式で出力することができます。この機能を使うためには、事前にDocker (https://docs.docker.jp/get-docker.html)のインストールが必要です。

`--preflight-option` オプションを指定すると、この処理を実行するpress-ready (https://github.com/vibranthq/press-ready)に対してオプションを追加できます。

```
# グレースケール化して出力
vivliostyle build manuscript.md --preflight press-ready --preflight-option gray-scale
# フォントを強制的にアウトライン化して出力
vivliostyle build manuscript.md --preflight press-ready --preflight-option enforce-outline
```

また、`--preflight press-ready-local` オプションを指定すると、PDF/X-1a形式への出力をローカル環境で実行します。ただし、通常はDocker環境上で実行することをおすすめします。

Dockerを利用した生成

- Example: render-on-docker (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/render-on-docker)

`vivliostyle build` コマンドで `--render-mode docker` オプションを指定すると、PDF出力時の環境としてDockerを指定できます（上記のオプションでは後処理のみDocker上で実行しますが、このオプションは全ての処理をDocker上で実行します）Dockerを用いることで出力時の環境を固定できるため、異なる環境・OSでも同じ出力結果となることを保証できます。

Docker render modeを使用する際は、以下の点に注意してください。

- Dockerはホスト環境から隔離されているため、ホストにインストールされているフォントを利用することができません。Dockerコンテナで標準で使用できるフォントは限られており、通常はローカルのフォントファイルを配置してCSSで指定するか、Google FontsなどのWebフォントを使用する必要があります。
- Dockerにマウントされるファイルはプロジェクトのworkspace directory（通常は `vivliostyle.config.js` を含むディレクトリ）のみで、その他のファイルはDockerコンテナ内部から参照することができない。イメージなどドキュメント内で参照されるファイルは全てworkspace directoryに含める必要があります。

PDFの「しおり」(Bookmarks)の生成

`vivliostyle build` コマンドで出力されるPDFには、目次の内容が「しおり」(PDF Bookmarks)として生成されます。PDFの「しおり」は、Adobe AcrobatのようなPDF閲覧ソフトで目次ナビゲーションに利用できます。

この「しおり」生成機能は、出版物に目次が含まれるときに有効になります。EPUBからPDFを生成する場合には、EPUBに含まれる目次が使われます。それ以外については目次の作成を参照してください。

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。*

Viteをベースとしたフレームワークを利用する

Vivliostyle CLIはWebフロントエンド開発ツールのViteをベースに開発されています。そのため、Viteをベースとしたフレームワークと組み合わせて使用できます。

Vivliostyle CLIは、フレームワークのタイプに応じて以下の2つの使用方法を提供します。

1. Viteプラグインを利用する

Vivliostyle CLIでは、ViteやRollupのプラグインを利用できます。`vite`オプションにプラグインの設定やViteの設定を記述します。

```
{
  vite: {
    plugins: [...],
  },
}
```

Vivliostyle Viewerは通常のWebアプリケーションとは異なり、JavaScriptの実行に制約があります。以下の点に注意してください。

- Vivliostyle Viewerはクライアントサイドのルーティング技術（シングルページアプリケーション; SPA）に対応していないため、各ページは静的にレンダリングする必要があります。`appType`に`mpa`を設定して動作しない環境では、この方法は使用できません。
- Vivliostyle Viewerはクライアントサイドでの HTML コンテンツの変更を検知できません。そのため、使用するUIフレームワークはサーバーサイドレンダリング（SSR）に対応している必要があります。クライアントサイドJavaScriptが含まれる場合、正しく動作しない可能性があります。

2. Vivliostyle CLIをViteプラグインとして使用する

- Example: with-astro (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/with-astro)
- Example: with-eleventy (https://github.com/vivliostyle/vivliostyle-cli/tree/main/examples/with-eleventy)

一部のフレームワーク（例: Astro (https://astro.build/)）は外部からViteプラグインを受け取れるものの、Viteプラグインとしては利用できません。この場合、Vivliostyle CLIをViteプラグインとして使用します。

例えば、Astroでは`astro.config.js`の`vite.plugins`オプションにVivliostyle CLIのViteプラグインを渡すことで、AstroとVivliostyle CLIを同時に利用できます。以下の例では、`openViewer: true`オプションを指定し、Astroのdevサーバー起動時にVivliostyle Viewerを自動で開くようにしています。

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

export default defineConfig({
  vite: {
    plugins: [
      createVitePlugin({
        openViewer: true,
      }),
    ],
  },
});
```

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()

> ***build***(`options`): `Promise` < `void` >

Build publication file(s) from the given configuration.

```
import { build } from '@vivliostyle/cli';
build({
  configPath: './vivliostyle.config.js',
  logLevel: 'silent',
});
```

Parameters

options

author?

`string` = `...`

bleed?

`string` = `...`

browser?

`string` = `...`

cmyk?

`boolean` | { `mapOutput?`: `string`; `overrideMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `reserveMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `warnUnmapped?`: `boolean`; } = `CmykSchema`

config?

`string` = `...`

configData?

`VivliostyleConfigSchema` | `null` = `...`

createConfigFileOnly?

`boolean` = `...`

cropMarks?

`boolean` = `...`

cropOffset?

`string` = `...`

css?

`string` = `...`

cwd?

`string` = `...`

disableServerStartup?

`boolean` = `...`

enableStaticServe?

`boolean` = `...`

enableViewerStartPage?

`boolean` = `...`

executableBrowser?

`string` = `...`

host?

`string` | `boolean` = `...`

ignoreHttpsErrors?

`boolean` = `...`

image?

`string` = `...`

input?

`string` = `...`

installDependencies?

`boolean` = `...`

language?

`string` = `...`

logger?

`LoggerInterface` = `...`

logLevel?

`"info"` | `"silent"` | `"verbose"` | `"debug"` = `...`

openViewer?

`boolean` = `...`

output?

`string` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

port?

`number` = `...`

preflight?

`"press-ready"` | `"press-ready-local"` = `...`

preflightOption?

`string` | `string`[] = `...`

pressReady?

`boolean` = `...`

projectPath?

`string` = `...`

proxyBypass?

`string` = `...`

proxyPass?

`string` = `...`

proxyServer?

`string` = `...`

proxyUser?

`string` = `...`

quick?

`boolean` = `...`

readingProgression?

`"ltr"` | `"rtl"` = `...`

renderMode?

`"local"` | `"docker"` = `...`

sandbox?

`boolean` = `...`

signal?

`AbortSignal` = `...`

singleDoc?

`boolean` = `...`

size?

`string` = `...`

stderr?

`Writable` = `...`

stdin?

`Readable` = `...`

stdout?

`Writable` = `...`

style?

`string` = `...`

template?

`string` = `...`

theme?

`string` | `false` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

timeout?

`number` = `...`

title?

`string` = `...`

userStyle?

`string` = `...`

viewer?

`string` = `...`

viewerParam?

`string` = `...`

vite?

`UserConfig` = `...`

viteConfigFile?

`string` | `boolean` = `...`

Returns

`Promise` < `void` >

---

create()

> ***create***(*`options`*): *`Promise`* < *`void`* >

Scaffold a new Vivliostyle project.

Parameters

options

author?

`string` = `...`

bleed?

`string` = `...`

browser?

`string` = `...`

cmyk?

`boolean` | { `mapOutput?`: `string`; `overrideMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `reserveMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `warnUnmapped?`: `boolean`; } = `CmykSchema`

config?

`string` = `...`

configData?

`VivliostyleConfigSchema` | `null` = `...`

createConfigFileOnly?

`boolean` = `...`

cropMarks?

`boolean` = `...`

cropOffset?

`string` = `...`

css?

`string` = `...`

cwd?

`string` = `...`

disableServerStartup?

`boolean` = `...`

enableStaticServe?

`boolean` = `...`

enableViewerStartPage?

`boolean` = `...`

executableBrowser?

`string` = `...`

host?

`string` | `boolean` = `...`

ignoreHttpsErrors?

`boolean` = `...`

image?

`string` = `...`

input?

`string` = `...`

installDependencies?

`boolean` = `...`

language?

`string` = `...`

logger?

`LoggerInterface` = `...`

logLevel?

`"info"` | `"silent"` | `"verbose"` | `"debug"` = `...`

openViewer?

`boolean` = `...`

output?

`string` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

port?

`number` = `...`

preflight?

`"press-ready"` | `"press-ready-local"` = `...`

preflightOption?

`string` | `string`[] = `...`

pressReady?

`boolean` = `...`

projectPath?

`string` = `...`

proxyBypass?

`string` = `...`

proxyPass?

`string` = `...`

proxyServer?

`string` = `...`

proxyUser?

`string` = `...`

quick?

`boolean` = `...`

readingProgression?

`"ltr"` | `"rtl"` = `...`

renderMode?

`"local"` | `"docker"` = `...`

sandbox?

`boolean` = `...`

signal?

`AbortSignal` = `...`

singleDoc?

`boolean` = `...`

size?

`string` = `...`

stderr?

`Writable` = `...`

stdin?

`Readable` = `...`

stdout?

`Writable` = `...`

style?

`string` = `...`

template?

`string` = `...`

theme?

`string` | `false` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

timeout?

`number` = `...`

title?

`string` = `...`

userStyle?

`string` = `...`

viewer?

`string` = `...`

viewerParam?

`string` = `...`

vite?

`UserConfig` = `...`

viteConfigFile?

`string` | `boolean` = `...`

Returns

`Promise` < `void` >

---

createVitePlugin()

> ***createVitePlugin***(*`inlineConfig`*): *`Promise`* < *`Plugin`* < *`any`* >*[]*>

Parameters

inlineConfig

author?

`string` = `...`

bleed?

`string` = `...`

browser?

`string` = `...`

cmyk?

`boolean` | { `mapOutput?` : `string` ; `overrideMap?` : [`string` | { `b` : `number` ; `g` : `number` ; `r` : `number` ; }, { `c` : `number` ; `k` : `number` ; `m` : `number` ; `y` : `number` ; }][]; `reserveMap?` : [`string` | { `b` : `number` ; `g` : `number` ; `r` : `number` ; }, { `c` : `number` ; `k` : `number` ; `m` : `number` ; `y` : `number` ; }][]; `warnUnmapped?` : `boolean` ; } = `CmykSchema`

config?

`string` = `...`

configData?

`VivliostyleConfigSchema` | `null` = `...`

createConfigFileOnly?

`boolean` = `...`

cropMarks?

`boolean` = `...`

cropOffset?

`string` = `...`

css?

`string` = `...`

cwd?

`string` = `...`

disableServerStartup?

`boolean` = `...`

enableStaticServe?

`boolean` = `...`

enableViewerStartPage?

`boolean` = `...`

executableBrowser?

`string` = `...`

host?

`string` | `boolean` = `...`

ignoreHttpsErrors?

`boolean` = `...`

image?

`string` = `...`

input?

`string` = `...`

installDependencies?

`boolean` = `...`

language?

`string` = `...`

logger?

`LoggerInterface` = `...`

logLevel?

`"info"` | `"silent"` | `"verbose"` | `"debug"` = `...`

openViewer?

`boolean` = `...`

output?

`string` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

port?

`number` = `...`

preflight?

`"press-ready"` | `"press-ready-local"` = `...`

preflightOption?

`string` | `string` [] = `...`

pressReady?

`boolean` = `...`

projectPath?

`string` = `...`

proxyBypass?

`string` = `...`

proxyPass?

`string` = `...`

proxyServer?

`string` = `...`

proxyUser?

`string` = `...`

quick?

`boolean` = `...`

readingProgression?

`"ltr"` | `"rtl"` = `...`

renderMode?

`"local"` | `"docker"` = `...`

sandbox?

`boolean` = `...`

signal?

`AbortSignal` = `...`

singleDoc?

`boolean` = `...`

size?

`string` = `...`

stderr?

`Writable` = `...`

stdin?

`Readable` = `...`

stdout?

`Writable` = `...`

style?

`string` = `...`

template?

`string` = `...`

theme?

`string` | `false` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

timeout?

`number` = `...`

title?

`string` = `...`

userStyle?

`string` = `...`

viewer?

`string` = `...`

viewerParam?

`string` = `...`

vite?

`UserConfig` = `...`

viteConfigFile?

`string` | `boolean` = `...`

Returns

`Promise` < `Plugin` < `any` >[]>

---

defineConfig()

> ***defineConfig***(`config`): `VivliostyleConfigSchema`

Define the configuration for Vivliostyle CLI.

Parameters

config

`VivliostyleConfigSchema`

Returns

`VivliostyleConfigSchema`

---

preview()

> ***preview***(*`options`*): *`Promise`* < *`ViteDevServer`* >

Open a browser for previewing the publication.

Parameters

options

author?

`string` = `...`

bleed?

`string` = `...`

browser?

`string` = `...`

cmyk?

`boolean` | { `mapOutput?` : `string` ; `overrideMap?` : [`string` | { `b` : `number` ; `g` : `number` ; `r` : `number` ; }, { `c` : `number` ; `k` : `number` ; `m` : `number` ; `y` : `number` ; }][]; `reserveMap?` : [`string` | { `b` : `number` ; `g` : `number` ; `r` : `number` ; }, { `c` : `number` ; `k` : `number` ; `m` : `number` ; `y` : `number` ; }][]; `warnUnmapped?` : `boolean` ; } = `CmykSchema`

config?

`string` = `...`

configData?

`VivliostyleConfigSchema` | `null` = `...`

createConfigFileOnly?

`boolean` = `...`

cropMarks?

`boolean` = `...`

cropOffset?

`string` = `...`

css?

`string` = `...`

cwd?

`string` = `...`

disableServerStartup?

`boolean` = `...`

enableStaticServe?

`boolean` = `...`

enableViewerStartPage?

`boolean` = `...`

executableBrowser?

`string` = `...`

host?

`string` | `boolean` = `...`

ignoreHttpsErrors?

`boolean` = `...`

image?

`string` = `...`

input?

`string` = `...`

installDependencies?

`boolean` = `...`

language?

`string` = `...`

logger?

`LoggerInterface` = `...`

logLevel?

`"info"` | `"silent"` | `"verbose"` | `"debug"` = `...`

openViewer?

`boolean` = `...`

output?

`string` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

port?

`number` = `...`

preflight?

`"press-ready"` | `"press-ready-local"` = `...`

preflightOption?

`string` | `string` [] = `...`

pressReady?

`boolean` = `...`

projectPath?

`string` = `...`

proxyBypass?

`string` = `...`

proxyPass?

`string` = `...`

proxyServer?

`string` = `...`

proxyUser?

`string` = `...`

quick?

`boolean` = `...`

readingProgression?

`"ltr"` | `"rtl"` = `...`

renderMode?

`"local"` | `"docker"` = `...`

sandbox?

`boolean` = `...`

signal?

`AbortSignal` = `...`

singleDoc?

`boolean` = `...`

size?

`string` = `...`

stderr?

`Writable` = `...`

stdin?

`Readable` = `...`

stdout?

`Writable` = `...`

style?

`string` = `...`

template?

`string` = `...`

theme?

`string` | `false` | `object` & `object` | (`string` | `object` & `object`)[] = `...`

timeout?

`number` = `...`

title?

`string` = `...`

userStyle?

`string` = `...`

viewer?

`string` = `...`

viewerParam?

`string` = `...`

vite?

`UserConfig` = `...`

viteConfigFile?

`string` | `boolean` = `...`

Returns

`Promise`<`ViteDevServer`>

VFM()

> ***VFM***(*`options?`*, *`metadata?`*): *`Processor`*

Create Unified processor for Markdown AST and Hypertext AST.

Parameters

options?

`StringifyMarkdownOptions`

Options.

metadata?

`Metadata`

Returns

`Processor`

Unified processor.

Interfaces

StringifyMarkdownOptions

Option for convert Markdown to a stringify (HTML).

Properties

Property	Type	Description
`assignIdToFigcaption?`	`boolean`	Assign ID to figcaption instead of img/code.
`disableFormatHtml?`	`boolean`	Disable automatic HTML format.
`footnote?`	`FootnoteMode | { mode: "pandoc"; } | { body?: Properties | DpubBodyFactory; call?: Properties | DpubCallFactory; mode: "dpub"; } | { body?: Properties | GcpmBodyFactory; duplicatedCall?: Properties | GcpmDuplicatedCallFactory; mode: "gcpm"; }`	Footnote output mode. Default is `'pandoc'` (endnote section).
`hardLineBreaks?`	`boolean`	Add ` ` at the position of hard line breaks, without needing spaces.
`imgFigcaptionOrder?`	`"img-figcaption" | "figcaption-img"`	Order of img and figcaption elements in figure.
`language?`	`string`	Document language (ignored in partial mode).
`math?`	`boolean`	Enable math syntax.
`partial?`	`boolean`	Output markdown fragments.
`replace?`	`ReplaceRule[]`	Replacement handler for HTML string.
`style?`	`string | string[]`	Custom stylesheet path/URL.
`title?`	`string`	Document title (ignored in partial mode).

TemplateVariable

Extends

- `Omit`<`ParsedVivliostyleInlineConfig`, `"theme"`>

Properties

Property	Type
`author`	`string`
`bleed?`	`string`
`browser?`	`object`
`browser.tag?`	`string`
`browser.type`	`"chrome"` | `"chromium"` | `"firefox"`
`cliVersion`	`string`
`cmyk?`	`boolean` | { `mapOutput?`: `string`; `overrideMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `reserveMap?`: [`string` | { `b`: `number`; `g`: `number`; `r`: `number`; }, { `c`: `number`; `k`: `number`; `m`: `number`; `y`: `number`; }][]; `warnUnmapped?`: `boolean`; }
`config?`	`string`
`configData?`	`VivliostyleConfigSchema` | `null`
`coreVersion`	`string`
`createConfigFileOnly?`	`boolean`
`cropMarks?`	`boolean`
`cropOffset?`	`string`
`css?`	`string`
`cwd?`	`string`
`disableServerStartup?`	`boolean`
`enableStaticServe?`	`boolean`
`enableViewerStartPage?`	`boolean`
`executableBrowser?`	`string`
`host?`	`string | boolean`
`ignoreHttpsErrors?`	`boolean`

Property	Type
image?	string
input?	object
input.entry	string
input.format	InputFormat
installDependencies?	boolean
language	string
logger?	LoggerInterface
logLevel?	"info" | "silent" | "verbose" | "debug"
openViewer?	boolean
output?	object & object & object[]
port?	number
preflight?	"press-ready" | "press-ready-local"
preflightOption?	string[]
pressReady?	boolean
projectPath	string
proxyBypass?	string
proxyPass?	string
proxyServer?	string
proxyUser?	string
quick?	boolean
readingProgression?	"ltr" | "rtl"
renderMode?	"local" | "docker"
sandbox?	boolean

Property	Type
`signal?`	`AbortSignal`
`singleDoc?`	`boolean`
`size?`	`string`
`stderr?`	`Writable`
`stdin?`	`Readable`
`stdout?`	`Writable`
`style?`	`string`
`template?`	`string`
`theme?`	`string` | `object` & `object` |(`string` | `object` & `object`)[]
`themePackage?`	`VivliostylePackageJson`
`timeout?`	`number`
`title`	`string`
`userStyle?`	`string`
`viewer?`	`string`
`viewerParam?`	`string`
`vite?`	`UserConfig`
`viteConfigFile?`	`string` | `boolean`

Type Aliases

Metadata

> ***Metadata*** = *`object`*

Metadata from Frontmatter.

Properties

base?

> `optional` **base**: `Attribute[]`

Attributes of `<base>`.

body?

> `optional` **body**: `Attribute[]`

Attributes of `<body>`.

class?

> `optional` **class**: `string`

Value of `<html class="...">`.

custom?

> `optional` **custom**: `object`

A set of key-value pairs that are specified in `readMetadata` not to be processed as `<meta>`. The data types converted from Frontmatter's YAML are retained. Use this if want to add custom metadata with a third party tool.

Index Signature

[`key` : `string`]: `any`

dir?

> `optional` **dir**: `string`

Value of `<html dir="...">`. e.g. `ltr`, `rtl`, `auto`.

head?

> `optional` ***head***: `string`

`<head>...</head>`, reserved for future use.

html?

> `optional` ***html***: `Attribute` *[]*

Attributes of `<html>`. The `id`, `lang`, `dir`, and `class` specified in the root take precedence over the value of this property.

id?

> `optional` ***id***: `string`

Value of `<html id="...">`.

lang?

> `optional` ***lang***: `string`

Value of `<html lang="...">`.

link?

> `optional` ***link***: `Attribute` *[][]*

Attribute collection of `<link>`.

meta?

> `optional` ***meta***: `Attribute` *[][]*

Attribute collection of `<meta>`.

script?

> `optional` ***script***: `Attribute` *[][]*

Attribute collection of `<script>`.

style?

> `optional` ***style***: `string`

`<style>...</style>`, reserved for future use.

title?

> `optional` ***title***: `string`

Value of `<title>...</title>`.

vfm?

> `optional` ***vfm***: `VFMSettings`

VFM settings.

StructuredDocument

> ***StructuredDocument*** = `object`

See

https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md (https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md)

Properties

children

> ***children***: `StructuredDocument` *[]*

href

> ***href***: `string`

sections?

> `optional` ***sections***: `StructuredDocumentSection` *[]*

title

> ***title***: `string`

StructuredDocumentSection

> ***StructuredDocumentSection*** = `object`

See

https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md (https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md)

Properties

children

> ***children***: `StructuredDocumentSection` *[]*

headingHtml

> ***headingHtml***: `string`

headingText

> ***headingText***: `string`

href?

> `optional` ***href***: `string`

id?

> `optional` ***id***: `string`

level

> ***level***: `number`

VivliostyleConfigSchema

> ***VivliostyleConfigSchema*** = `BuildTask` *[]* | `BuildTask`

See

https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md (https://github.com/vivliostyle/vivliostyle-cli/blob/main/docs/config.md)

VivliostylePackageMetadata

> ***VivliostylePackageMetadata*** = `SchemaWithPipe`<readonly [`Omit`< `ObjectSchema`<{ `template`: `SchemaWithPipe`<readonly [`RecordSchema`< `SchemaWithPipe`<readonly [..., ..., ...]>, `ObjectSchema`<{ `description`: ...; `name`: ...; `prompt`: ...; `source`: ...; }, `undefined`>, `undefined`>, `TitleAction`<{[`key`: `string`]: `object`; }, `"VivliostyleTemplateMetadata"`>]>; `theme`: `SchemaWithPipe`<readonly [`ObjectSchema`<{ `author`: `SchemaWithPipe`<...>; `category`: `SchemaWithPipe`<...>; `name`: `SchemaWithPipe`<...>; `style`: `SchemaWithPipe`<...>; `topics`: `SchemaWithPipe`<...>; }, `undefined`>, `TitleAction`<{ `author?`: ... | ...; `category?`: ... | ...; `name?`: ... | ...; `style?`: ... | ...; `topics?`: ... | ...; }, `"VivliostyleThemeMetadata"`>]>; }, `undefined`>, `"entries"` | `"~types"` | `"~run"` | `"~standard"`> & `object`, `TitleAction`<{ `template?`: {[`key`: `string`]: `object`; }; `theme?`: { `author?`: `string`; `category?`: `string`; `name?`: `string`; `style?`: `string`; `topics?`: `string`[]; }; }, `"VivliostylePackageMetadata"`>]>

VivliostylePackageMetadata

> ***VivliostylePackageMetadata*** = `v.InferInput`<typeof `VivliostylePackageMetadata`>

Variables

readMetadata()

> `const` ***readMetadata***: (`md`, `customKeys?`) => `Metadata`

Read metadata from Markdown frontmatter.

Keys that are not defined as VFM are treated as `meta`. If you specify a key name in `customKeys`, the key and its data type will be preserved and stored in `custom` instead of `meta`.

Parameters

md

`string`

Markdown.

customKeys?

`string`[]

A collection of key names to be ignored by meta processing.

Returns

`Metadata`

Metadata.