冊子本の原稿として Markdown を使う
– LaTeX と Pandoc による事例から–

鹿野 桂一郎
ラムダノート株式会社
k16.shikano@lambdanote.com

2019 年 11 月 14 日
於 「アンテナハウス株式会社
Markdown+CSS/TeX **で冊子本を作ってみた」**

Markdown とは？

- 「プレーンテキストを整形」する記法
  - Markdown = テキストエディター上で何となく見た目を再現できる装飾付きのテキスト
- そのテキストを「HTML へ変換」するツール
  - Markdown = Free software
    （オリジナルは BSD 系ライセンス）

John Gruber による定義：
`https://daringfireball.net/projects/markdown/`

つまりMarkdownとは、
「記法」であり
「表現への変換ツール」である

Markdownは
実装者の数だけある！

私も実装したことがあります

- TEX による Markdown 処理系
- TEX のマクロ言語で実装されていて、Markdown の記法で書いた LATEX 形式の原稿から、そのまま PDF を作れる

```
\documentclass[uplatex]{jsarticle}
\usepackage{md}

\begin{document}
\begin{markdown}

# markdown-tex

markdown processor in TeX.

## supported syntax

So far, you can write normal paragraph, codelist, quo

As inline styles, `tt`, *italic*, and **bold** are av

## example

Inshort, you can write something like below.
Below is a `codeline` block.

    codeline
      another codenline
```

1 markdown-tex

markdown processor in TeX.

1.1 supported syntax

So far, you can write normal paragraph, c
As inline styles, `tt`, *italic*, and **bold** are a

1.2 example

Inshort, you can write something like bel

```
codeline
  another codenline
```

Here is a enumerate list.

LaTeX とは

- 「プレーンテキストを整形」する記法
  - LaTeX = テキストエディター上で文書の見た目を指示できる
- そのテキストを「PDF へ変換」するツール
  - LaTeX = Free software （LPPL ライセンス）

Markdown の定義と同じ！

つまりLaTeXは、
「記法」であり
「表現への変換ツール」である

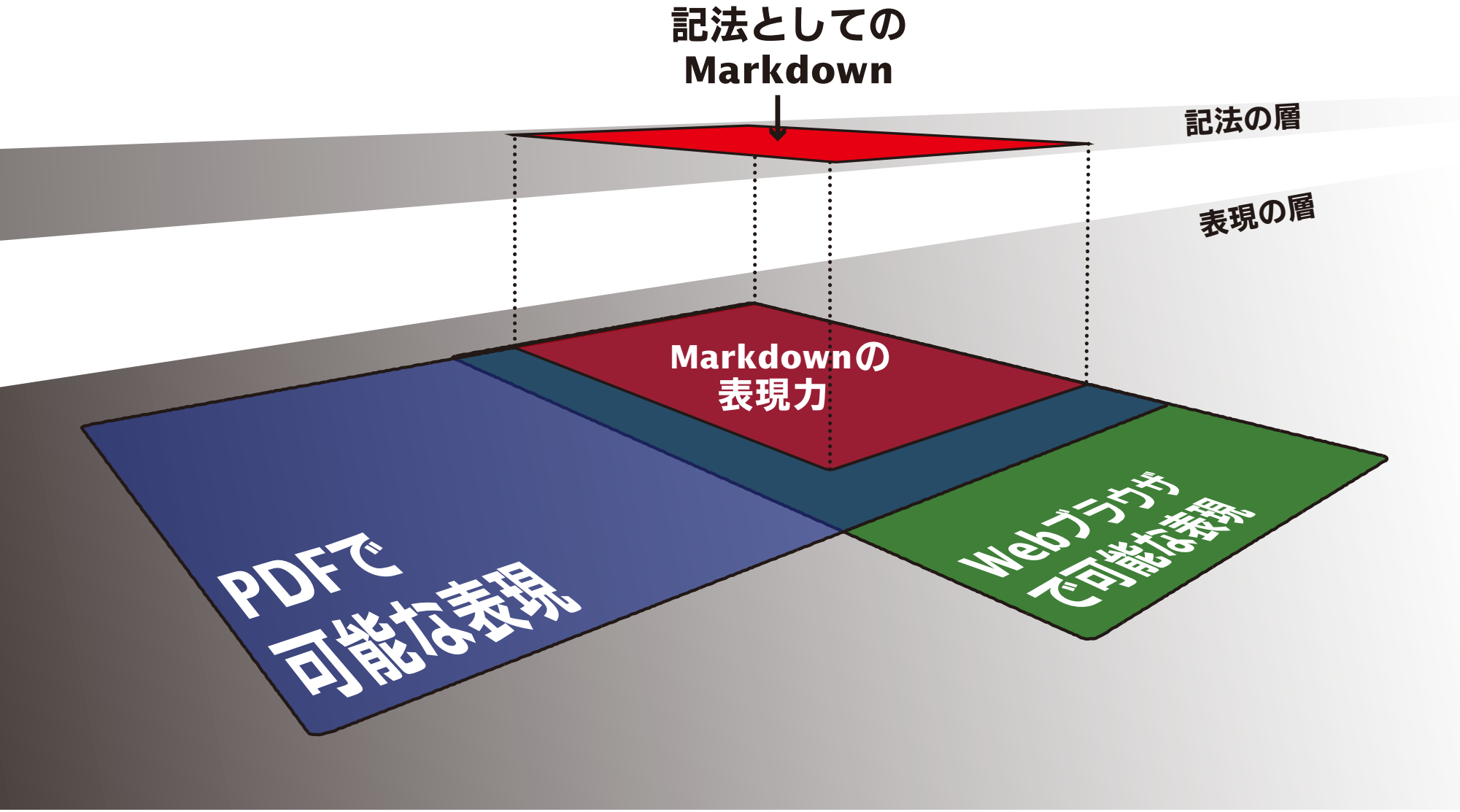
LaTeX と Markdown は、どちらも「記法」であり「表現への変換ツール」である

（構造化文章のことは忘れましょう）

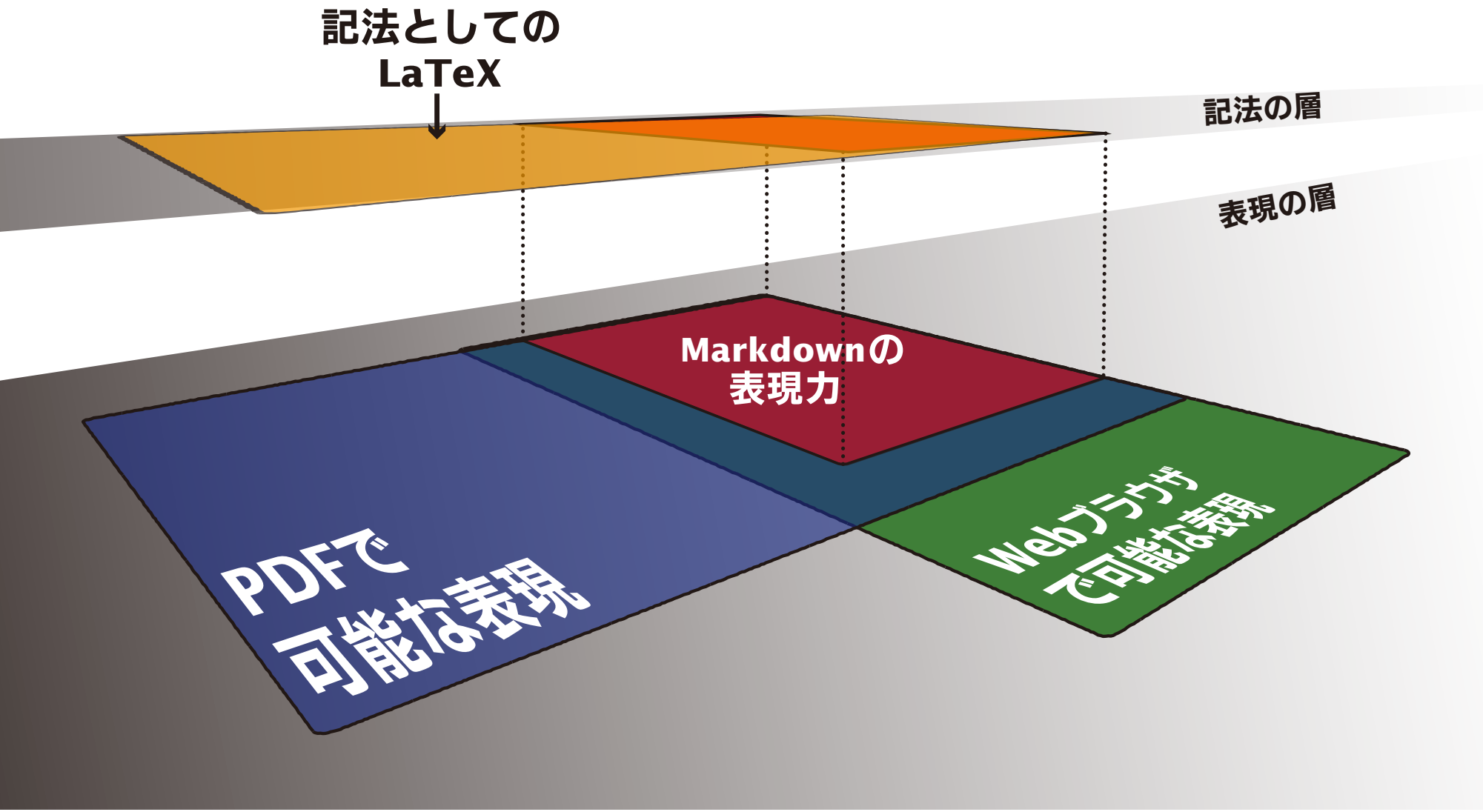
LaTeXではPDFを作れるので、Markdownの記法をLaTeXに変換さえすれば、PDFが作れる？

（Markdown から LaTeX への変換に対する動機）

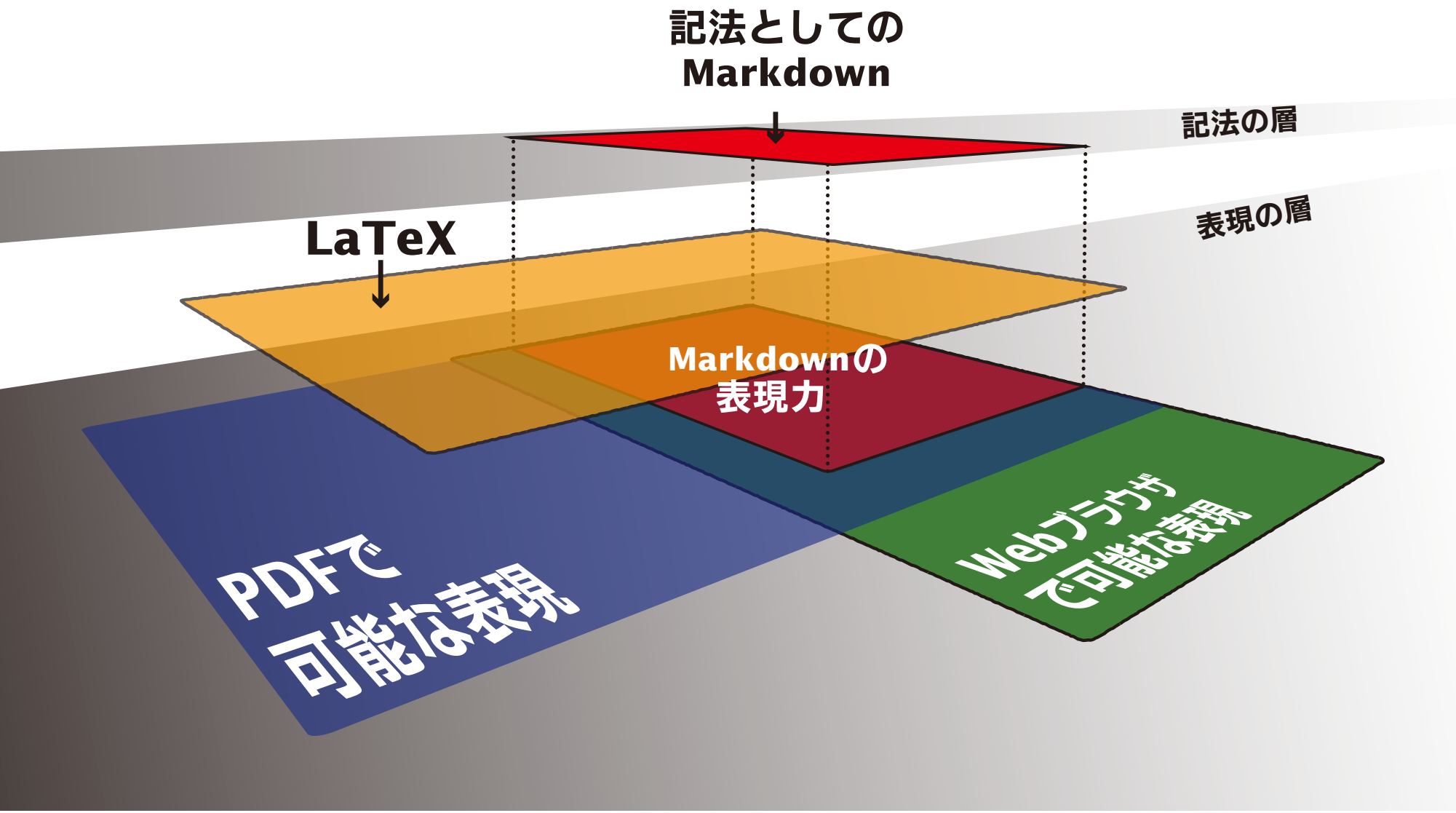
Markdownの表現力



LaTeX の表現力


記法としての
LaTeX
記法の層
表現の層
Markdownの
表現力
PDFで
可能な表現
Webブラウザ
で可能な表現

LaTeX を中間層とすることで Markdown からPDFの表現力を得る



Markdown と LaTeX の関係

- Markdown ＝
  LaTeX の標準的な記法の代わりとなる、より簡易な記法

- LaTeX ＝
  Markdown 記法のテキストから HTML ではなく PDF を出力したいときのバックグラウンド

Markdown と HTML の間の関係と、ほぼ同じ関係が成り立っている。

Markdown からPDF を作れる

$\neq$

Markdown から冊子本を作れる

Markdownの表現力では不足する点をLaTeXで直接補えば、冊子本のPDFを作れるか？

LaTeX の一部にMarkdown を使う方法

- 「冊子本のための PDF」に必要とされる構造だけならば、その選択肢もありえた
- 原稿に LaTeX の記法が混在することをどこまで許容できるか
- 電子版との原稿の分裂の可能性

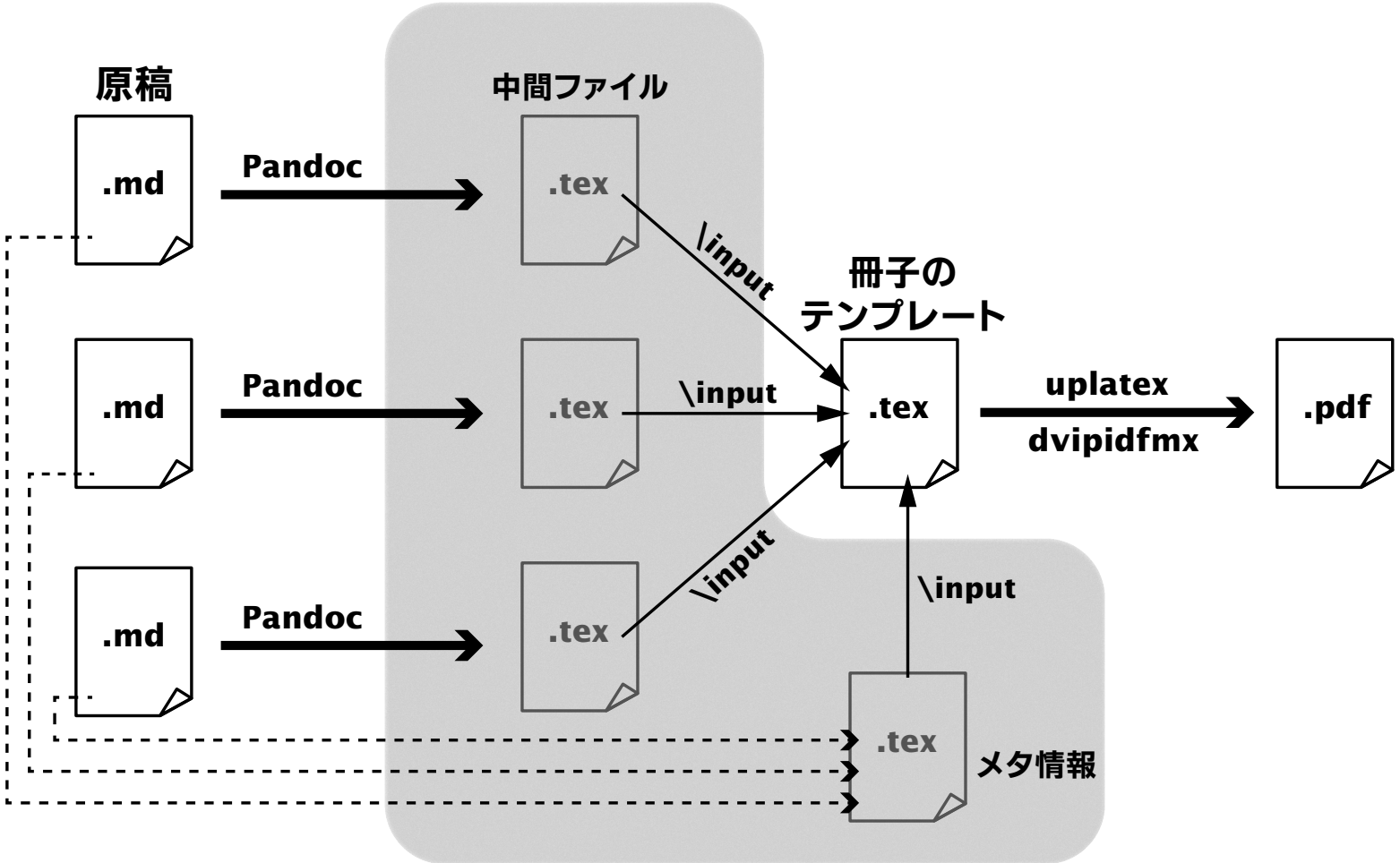
「XML 原稿の中に Markdown の島がある」ような原稿もあった（翻訳書）

そこでPandoc

- https://pandoc.org/
- **多様なドキュメント形式の相互変換ツール**
- Markdown **から** PDF **への変換にも対応**
- PDF **を出力する場合のバックグラウンドには** LaTeX **が採用されている**

ただし、これは一般向けのキャッチーな説明。Pandoc **の本懐は、多様なドキュメント形式を「**Markdown **の構造」でとらえるためのツールという点。**

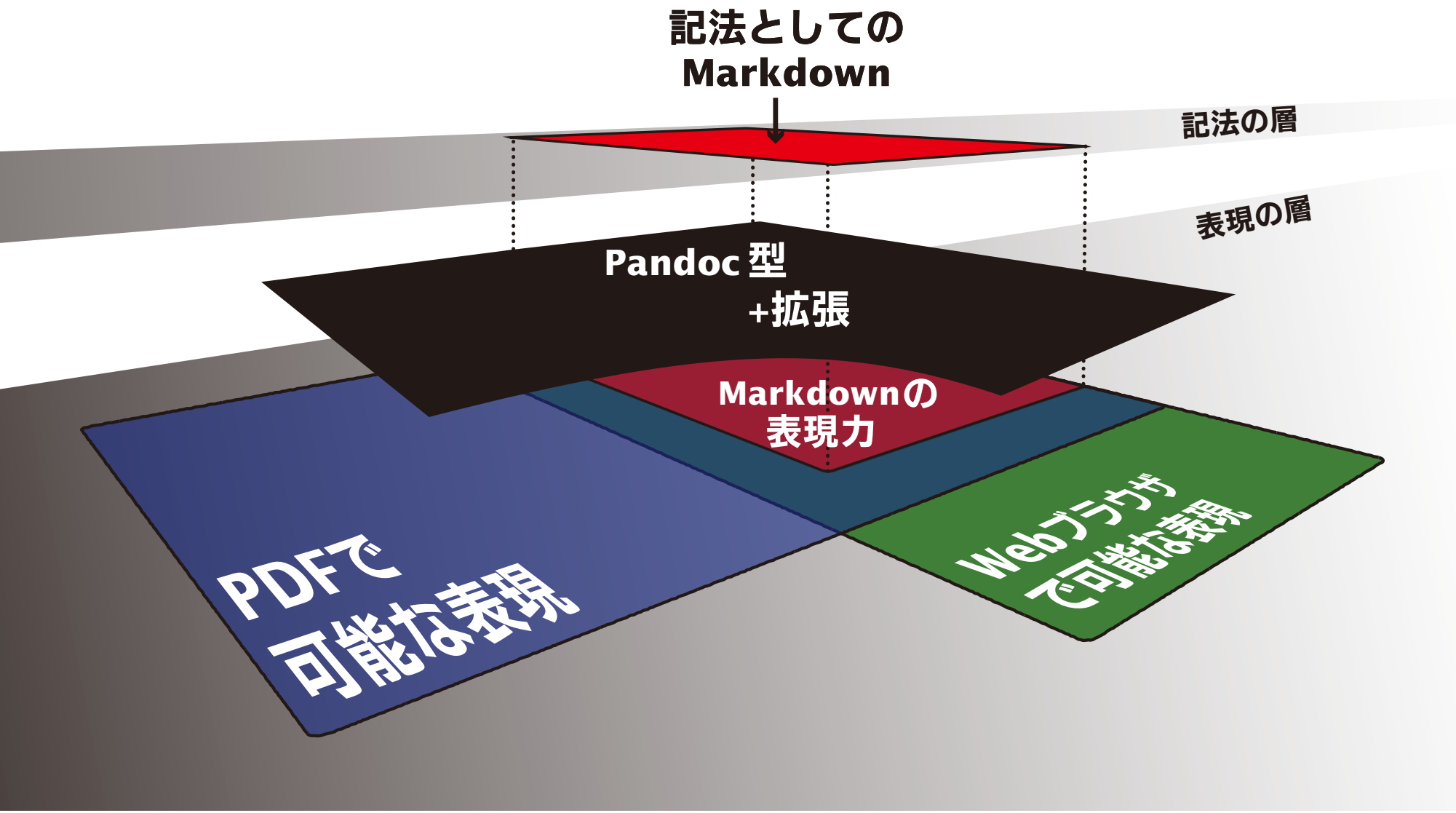
不定期刊行誌『n月刊ラムダノート』制作フロー



Pandoc を採用している理由

- Pandoc が提供する型は、Markdown の記法から自然に決まるドキュメントの構造
- その構造の基本的な構成要素は、冊子本にとっても十分
  - むしろ最低限の構造しかないので、見た目で原稿をごまかせる余地が減る
  - 参考：ラムダノートの記事執筆スタイルガイド：
    `https://gist.github.com/LambdaNote/0d33b7d8284a3c99cffd1a5aa83c115f`
- 足りない構造を補う、豊富なツールチェインと拡張性
  - 素の Markdown と親和性の高い「記法」がセットで提供されている（これが最も重要）

Pandoc を中間層とすることで Markdown から冊子本の表現力を得る



Pandoc が提供する型

data Block

Block element.

Constructors

- Plain [Inline]
- Para [Inline]
- LineBlock [[Inline]]
- CodeBlock Attr Text
- RawBlock Format Text
- BlockQuote [Block]
- OrderedList ListAttributes [[Block]]
- BulletList [[Block]]
- DefinitionList [([Inline], [[Block]])]
- Header Int Attr [Inline]
- HorizontalRule
- Table [Inline] [Alignment] [Double] [TableCell] [[TableCell]]
- Div Attr [Block]
- Null

data Inline

Inline elements.

Constructors

- Str Text
- Emph [Inline]
- Strong [Inline]
- Strikeout [Inline]
- Superscript [Inline]
- Subscript [Inline]
- SmallCaps [Inline]
- Quoted QuoteType [Inline]
- Cite [Citation] [Inline]
- Code Attr Text
- Space
- SoftBreak
- LineBreak
- Math MathType Text
- RawInline Format Text
- Link Attr [Inline] Target
- Image Attr [Inline] Target
- Note [Block]
- Span Attr [Inline]

出典：http://hackage.haskell.org/package/pandoc-types

冊子本で使えるPandocの拡張

- YAML Metadata Block
- pandoc-citeproc + CSL （参考文献）
- `{...}`構文による属性指定
  - pandoc-crossref （相互参照）
  - コードブロックのハイライト
- footnotes （脚注）
- Pipe Tables （表）
- Pandoc フィルター

YAML Metadata Block

- 冊子本の構成やタイトルなどの情報を、原稿ファイルに YAML 形式で埋め込める
- YAML が記法の一部になっているといえる

pandoc-citeproc + CSL （参考文献）

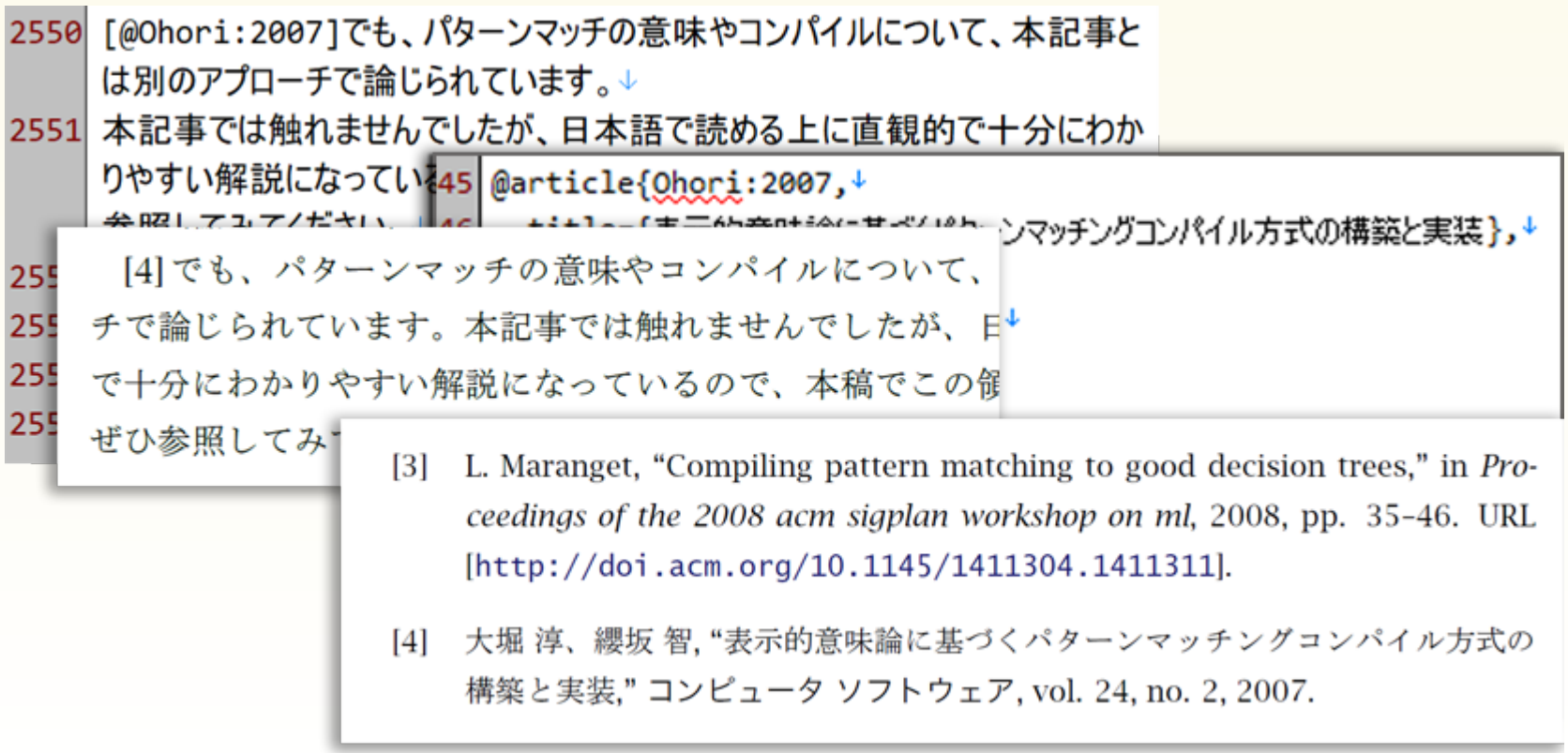
- BibTEX 形式で保存した参考文献が利用できる
- citation の記法は［@. . .］
- citation とエントリの整形には
  CSL（https://citationstyles.org/）を利用

```
2550 [@Ohori:2007]でも、パターンマッチの意味やコンパイルについて、本記事と
     は別のアプローチで論じられています。↓
2551 本記事では触れませんでしたが、日本語で読める上に直観的で十分にわか
     りやすい解説になってい
     参照してみてください。↓
2552 ↓
2553 \setbib↓
2554 ↓
2555 [EOF]
```

```
45 @article{Ohori:2007,↓
46 ␣␣title={表示的意味論に基づくパターンマッチングコンパイル方式の構築と実装},↓
47 ␣␣author={大堀␣淳、櫻坂␣智},↓
48 ␣␣journal={コンピュータ␣ソフトウェア},↓
49 ␣␣volume={24},↓
50 ␣␣number={2},↓
51 ␣␣pages={2_113-2_132},↓
52 ␣␣year={2007},↓
53 ␣␣doi={10.11309/jssst.24.2_113}↓
54 }↓
55 [EOF]
```

pandoc-citeproc + CSL （参考文献）

- BibTEX 形式で保存した参考文献が利用できる
- citation の記法は [@...]
- citation とエントリの整形には CSL（https://citationstyles.org/）を利用



pandoc-crossref （相互参照）

- 図、表、コードリスト、節などに ID を指定し、本文から番号で参照できるようにする機構
- ID の付与には属性の記法`{#...}`を使う
- 参照には citation と同じ記法 `[@...]` を使う
- LaTeX に任せるべきではない

```
961 本稿ではコンパイルの話には深く立ち入りませんが、このように複数の段階を経てコンパイルされるということは覚えておいてください（図[-@fig:smltoc]上）。
962
963 ![SMLからCに変換する際のコードとデータのフロー図](images/sml_to_c.png){#fig:smltoc}
964
965 以降では、パターンマッチのコンパイルを見る前に、代数的データ型のコンパイルについて説明します。
966 なお、代数的データ型については、段階を分けることなくそのままC言語のコードに対応付けられます（図[-@fig:smltoc]下）。
```

コードブロックのハイライト

- コードブロックに属性を指定することで、予約語のハイライトなどを有効にする機構
- 属性の記法{...}を使う
- LaTeX に任せることもできる

````
`printPathname` は、おおまかには[-@lst:printPathname-compiled]のように変換されます。
本当はパターンマッチ以外の処理も必要なので、コンパイラの内部の処理で型が付いたり名前が変わったりしますが、ここでは無視しています。

```{#lst:printPathname-compiled .c caption="printPathnameの変換後"}
void
printPathname(sml_string prefix, struct entry *e)
{
  swtich(e->tag) {
    case File: {
      sml_string n
      /* 処理 */
      break;
    }
```

printPathname は、おおまかにはリスト 1.30 のように変換されます。本当はパターンマッチ以外の処理も必要なので、コンパイラの内部の処理で型が付いたり名前が変わったりしますが、ここでは無視しています。

リスト 1.30 printPathname の変換後

```
void
printPathname(sml_string prefix, struct entry *e)
{
  swtich(e->tag) {
    case File: {
      sml_string name = e->data.file;
      /* 処理 */
      break;
    }
```
````

footnotes （脚注）

- **脚注のための簡便な記法**
- `[^aaa]` **および** `[^aaa]:  ...`

```
376 なお、パターンマッチはRubyの次期バージョンである2.7[^rubyversion]において、あくまでも実験的
    機能という位置づけでリリースされる予定となっています。
377 本稿で紹介する仕様は2019年10月にリリースされた2.7.0-preview2のものとなりますが、将来変
    更される可能性もあります。
378
379 [^rubyversion]: Rubyでは年
    ンアップリリースが行われます。前
    スが中心となります。2019年10月
```

なお、パターンマッチはRubyの次期バージョンである2.7[†4]において、あくまでも実験的機能という位置づけでリリースされる予定となっています。本稿で紹介する仕様は2019年10月にリリースされた2.7.0-preview2のものとなりますが、将来変更される可能性もあります。

[†4] Rubyでは年1回のマイナーバージョンアップリリースと年数回のティーニーバージョンアップリリースが行われます。前者は言語仕様変更を伴う大きめのアップデートで、後者はバグフィックスが中心となります。2019年10月現在、2.6系（2.6.5）が最新です。

Pipe tables （表）

- さまざまな表の記法のうち、表現力と記述性のバランスがよい
- 表のレンダリング結果を完全に制御できるわけではない

表 2.1: Rubyの代表的なオブジェクトとリテラル

データの種類	対応するクラス	リテラル例
整数	Integerクラス	0
文字列	Stringクラス	"a"
範囲	Rangeクラス	0..1 （0以上1以下の範囲を表す）
		0.. （0以上の範囲を表す）

```
166 Rubyではすべてのデータはオブ
167 以下の表␣@tbl:objlitera
168 ↓
169 |␣データの種類␣|␣␣対応するクラス␣␣|␣リテラル例␣|↓
170 |:-----------|:--------------|:---------|↓
171 |␣整数␣␣␣␣␣␣|␣`Integer`クラス␣|␣`0`␣␣␣␣␣|↓
172 |␣文字列␣␣␣␣|␣`String`クラス␣␣|␣`"a"`␣␣␣|↓
173 |␣範囲␣␣␣␣␣␣|␣`Range`クラス␣␣|␣`0..1`␣(`0`以上`1`以下の範囲を表す)␣|↓
174 |␣␣␣␣␣␣␣␣␣␣␣|␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣|␣`0..`␣␣(`0`以上の範囲を表す)␣|↓
175 ↓
176 Table:␣Rubyの代表的なオブジェクトとリテラル␣{#tbl:objliteral}↓
```

Pandoc フィルター

- Pandoc 型の値を直接、プログラムにより操作できる
- 「本文内の行コメント」のような仕組みを導入することも可能に

```
17 main␣::␣IO␣()↓
18 main␣=␣toJSONFilter␣inline↓
19 ↓
20 inline␣::␣Inline␣->␣Inline↓
21 inline␣(Str␣s)␣=␣sharp␣$␣kappa␣(Str␣s)↓
22 inline␣(Code␣attr␣s)␣=␣sharp␣$␣kappa␣(Code␣attr␣s)↓
23 inline␣(Span␣attr␣xs)␣=␣Span␣attr␣$␣map␣inline␣xs↓
24 inline␣x␣=␣x↓
25 ↓
26 kappa␣s␣=␣subRegexInRawinline␣(R.mkRegex␣"[κ]")␣s␣"¥¥ensuremath{¥¥mathtt{¥¥kappa}}"↓
27 sharp␣s␣=␣subRegexInRawinline␣(R.mkRegex␣"SML＃")␣s␣"¥¥texorpdfstring{SML¥¥ensuremath{
```

それでもLaTeXの知識は必要

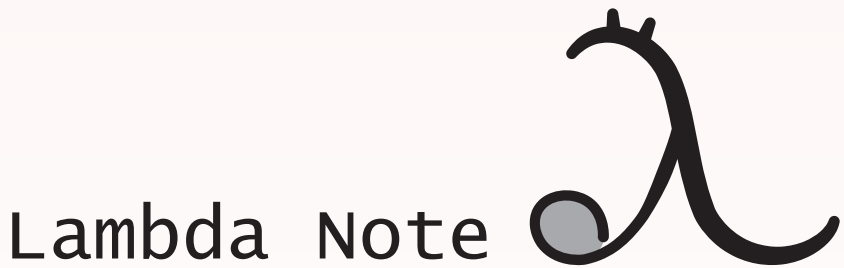
- 冊子本の全体構成を組み立てるには、Pandocのテンプレート機構か、それに類する仕組みが必要になる
- 「最後の追い込み」が必要
- エスケープの問題
- 数式の記法は？
- 索引の記法は？
- 生のLaTeX記法を埋め込めるか
  - 「構造」ないと埋め込みにくい
  - MarkdownにHTMLを埋め込めるのは、HTMLが記法であり「構造」でもあるから

まとめ

- Markdown はドキュメントの構造というより、むしろ記法
- したがって Markdown 原稿から冊子本（のための PDF）を作る際も、構造にスタイルを当てはめるという XML 的な考え方でなく、むしろ構造を暗に表す記法そのものが問題になる
- Pandoc は、Markdown の記法により決まる構造を利用しやすくすると同時に、Markdown 自体に欠けている冊子本で必要となる記法も提供してくれている

宣伝

- ラムダノート株式会社は出版を中心として技術文書まわりのお手伝いをいろいろする会社です
  - https://lambdanote.com



おまけ：Markdown + CSSの方向性？