

# VScode でレポートを書こう! - 準備編 -

山根義琉  
情報電子工学科 2 年

2026 年 01 月 08 日

## 目次

1 VScode の基礎知識	2
1.1 拡張機能	2
1.2 設定	2
1.3 スペニット	2
2 色々な仕組み	2
2.1 MarkDown	2
2.2 ノベルライター	2
2.3 LaTeX	3
2.4 Typst	3
2.5 結局何が良い?	3
3 環境設定	3
3.1 MarkDown	3
3.1.1 手順	3
3.2 ノベルライター	3
3.2.1 手順	3
3.3 LaTeX	4
3.3.1 手順	4
3.4 Typst	6
3.4.1 手順	6
4 実例	6
4.1 MarkDown でレポート	6
4.1.1 ソースコード	6
4.1.2 PDF 出力	6
4.2 Typst でレポート	7
4.2.1 ソースコード	7
4.2.2 PDF 出力	7
4.3 LaTeX でレポート	8
4.3.1 ソースコード	8
4.3.2 PDF 出力	8

# 1 VScode の基礎知識

VScode は IDE<sup>1</sup>であり、テキストベースのファイルならばほぼすべての編集が快適に行える。ここではその機能の一部と共に VScode での GUI<sup>2</sup> 上の操作方法について紹介し、ドキュメント作成の準備を行う。この PDF は VScode で typst を使い書いている。

## 1.1 拡張機能

VScode では機能を拡張するために主に個人が開発している「拡張機能」をインストールすることができる。またそれについて「設定」を行うことでカスタマイズすることが可能である。

## 1.2 設定

設定にはグローバルとワークスペースがあり前者は如何様なフォルダーを開いても全てにおいて適用される設定を記述するもので、後者はそのフォルダを編集する際に適用される設定のみを記述できる。JSON というファイルを使い設定を記述することができる。ファイルはこの位置に置けばいい。



設定画面から GUI でも設定は変更できる。JSON ファイルの例はこんな感じ。

```
"files.autoSave": "afterDelay",
"[typst)": {
    "editor.wordWrap": "on",
}
```

この JSON の中身は使う組版<sup>3</sup>エンジンによって色々な設定があるのでその都度紹介する。

## 1.3 スペニット

スペニットというファイルを編集することである単語を発動条件にして雛形を出力することが可能となる。これはグローバルと拡張子<sup>4</sup>ごとに設定できる。便利なのでぜひ使ってみてほしい。

# 2 色々な仕組み

作成したいドキュメントの種類ごとに最適な方法があるので以降、個人的に一押しした4つを紹介する。

## 2.1 MarkDown

一番ベーシックな文章作成方式で様々な場面で使われる。とりあえず MarkDown さえ入れておけば何かしらができる。

### 良い点

- シンプル
- コンパイルなし
- 互換性が高い

### 悪い点

- 応用が効かない(組版がない)
- 外部ライブラリがない
- 少し環境依存する

## 2.2 ノベライター

この組み合わせは原稿用紙や進捗管理が恐ろしく有能なのでベタ打ちのテキストを作成するときは非常に強力。

### 良い点

- 日本語に特化
- 文章を書くことに有用な機能
- 文字数カウントから PDF 化まで大抵できる

### 悪い点

<sup>1</sup>総合開発環境

<sup>2</sup>graphical user interface

<sup>3</sup>PDF などにレンダリングする仕組み

<sup>4</sup>.以降のやつ eg .txt,.tex,.py

- 記号や数式はほとんど機能しない
- カスタムできない
- 互換性がない

## 2.3 LaTeX

TeX から派生した近代的な大型の組版システムで圧倒的な歴史と知名度を誇る。

### 良い点

- 今まで上で挙げたこともそれ以外もすべてできる
- 数式・図・表・絵まで同じファイルで書ける
- 大抵のことは自動でできる

### 悪い点

- 独自性が高く難解
- エラーが不親切
- コンパイルが必要(超遅い)

## 2.4 Typst

現時点で一番有力な LaTeX の後継候補で Rust 製のモダンな組版システム。ナウい。

### 良い点

- 導入・エラー・コンパイルが初心者に優しい
- コンパイルがほぼノータイム
- 文法が明確でプログラミング言語より
- MarkDown 記法を模倣した視認性の高い方
- 式

### 悪い点

- テンプレートが少ない
- LaTeX の知識をつけても活かせない

## 2.5 結局何が良い?

まず MarkDown を触って慣れることが大切なのでそこから。文章特化ならばノベルライターへ、レポート(組版)特化なら Typst が無難。LaTeX も超おすすめなのだが難解でおそらく挫折するので試したい人だけにした方が良い。

## 3 環境設定

この2つは特に厄介である。複雑な環境構築とその他諸々の設定をしなくてはならない。

### 3.1 Markdown

基礎情報	No. 2
数式	○
図表	△
描画	×
拡張子	.md
コンパイル	なし

#### 3.1.1 手順

- 拡張機能 Markdown All in One と markdownlint をインストール
- .md ファイルを作成
- 右上のボタンでプレビューとかする

### 3.2 ノベルライター

基礎情報	No. 3
数式	×
図表	×
描画	×
拡張子	.txt
コンパイル	なし

#### 3.2.1 手順

- 拡張機能 novel-writer をインストール
- .txt ファイルを作成
- 右上のボタンでプレビューとかする

### 3.3 LaTeX

基礎情報

No. 4

- 数式 ○
- 図表 ○
- 描画 ○
- 拡張子 .tex
- コンパイル あり

#### 3.3.1 手順

1. ターミナルで TeX Live をインストール

```
brew install --cask mactex-no-gui
exec $SHELL -l
sudo tlmgr update --self --all
```

ここで `-no-gui` としているが VScode で LaTeX を使わない場合はその部分を消したコマンドにする
2. path を通す
3. 拡張機能 LaTeX Workshop をインストール
4. settings.json に以下を記述

```
//--LaTeXの整形--//
"latex-workshop.formatting.latex": {
  "latexindent",
//--LaTeXエンジンの設定--//
"latex-workshop.latex.outDir": "output",
"latex-workshop.latex.tools": [
  { //--cmdxがあった
    "name": "lualatex",
    "command": "lualatex",
    "args": [
      "-file-line-error",
      "-synctex=1",
      "-interaction=nonstopmode",
      "-halt-on-error",
      "-output-directory=%OUTDIR%",
      "-shell-escape", //svg用
      "%DOC%"
    ],
    "env": {}
  },
  {
    "name": "pbibtex(ja)",
    "command": "pbibtex",
    "args": [
      "-kanji=utf8",
      "%OUTDIR%/%DOCFILE%"
    ]
  }
],
{
  "name": "biber",
  "command": "biber",
  "args": [
    "--output-directory=%OUTDIR%",
    "%DOCFILE%"
  ]
},
],
//--LaTeXレシピの設定--//
"latex-workshop.latex.recipes": [
{
  "name": "lua",
  "tools": [
    "lualatex",
  ]
},
{
  "name": "lua*2",
  "tools": [
    "lualatex",
    "lualatex",
  ]
},
{
  "name": "lua-bib-lua*2",
  "tools": [
    "lualatex",
    "pbibtex(ja)",
    "lualatex",
    "lualatex",
  ]
},
{
  "name": "lua-biber-lua*2",
  "tools": [
    "lualatex",
    "biber",
    "lualatex",
    "lualatex",
  ]
},
]
```

```
"command": "pbibtex",
"args": [
  "-kanji=utf8",
  "%OUTDIR%/%DOCFILE%"
]
},
{
  "name": "biber",
  "command": "biber",
  "args": [
    "--output-directory=%OUTDIR%",
    "%DOCFILE%"
  ]
},
],
//--LaTeXレシピの設定--//
"latex-workshop.latex.recipes": [
{
  "name": "lua",
  "tools": [
    "lualatex",
  ]
},
{
  "name": "lua*2",
  "tools": [
    "lualatex",
    "lualatex",
  ]
},
{
  "name": "lua-bib-lua*2",
  "tools": [
    "lualatex",
    "pbibtex(ja)",
    "lualatex",
    "lualatex",
  ]
},
{
  "name": "lua-biber-lua*2",
  "tools": [
    "lualatex",
    "biber",
    "lualatex",
    "lualatex",
  ]
},
]
```

```

1,
//--コンパイル時--//
// % + sでファイル保存時に自動でコンパイルする
"latex-workshop.latex.autoBuild.run":
"onSave",
// 直近で使用したレシピでコンパイルする
"latex-workshop.latex.recipe.default":
"lastUsed",
// LaTeX Workshopのインテリセンスを有効にする
"latex-
workshop.intellisense.package.enabled":
true,
// PDFをVSCode で開く
"latex-workshop.view.pdf.viewer": "tab",
// PDFの幅を調整する
"latex-workshop.view.pdf.zoom": "page-
width",
// PDFを右側に開く
"latex-
workshop.view.pdf.tab.editorGroup":
"right",
//不要なファイルを削除
"latex-workshop.latex.autoClean.run":
"onBuilt",
"latex-
workshop.latex.autoBuild.onSave.files.ignore":
[
  "**/*.{sty}",
  "**/*.{cls}",
  "**/*.{tex}"
],
"latex-workshop.latex.clean.fileTypes": [
  "*.{aux}",
  "*.{bbt}",
  "*.{blg}",
  "*.{idx}",
  "*.{ind}",
  "*.{lof}",
  "*.{lot}",
  "*.{out}",
  "*.{toc}",
  "*.{acn}",
  "*.{acr}",
  "*.{alg}",
  "*.{glg}",
  "*.{glo}",
  "*.{gls}",
  "*.{ist}"
]

```

```

"*.fls",
"*.log",
"*.fdb_latexmk",
"*.snm",
"*.nav",
"*.dvi",
"*.synctex.gz",
],

```

#### 4. ワークスペースの setting.json を設定する (自由にして良い)

```

// オートセーブ
"files.autoSave": "afterDelay",
// LaTeXファイルのときに文字列を折り返す
"[latex)": {
  "editor.wordWrap": "on",
},
// bibTeXファイルのときに文字列を折り返さない
"[bibtex)": {
  "editor.wordWrap": "off",
},
// 実行時のレシピを指定
"latex-workshop.latex.recipe.default":
"lua_bibla",
// pdfの見た目を自由に変更できる
"latex-
workshop.view.pdf.color.dark.pageBorderColor":
"lightgrey",
"latex-
workshop.view.pdf.color.dark.backgroundColor":
"#0f1419",
"latex-
workshop.view.pdf.color.dark.pageColorsBackground":
"#FFFDE8",
"latex-
workshop.view.pdf.color.dark.pageColorsForeground":
"#102010",

```

#### 5. メニューバーの TeX アイコンから実行(セー ブ時に勝手にコンパイルする設定は時に邪 魔なのでなくして良い)

## 3.4 Typst

### 基礎情報

No. 5

- 数式 ◎
- 図表 ○
- 描画 ◎
- 拡張子 .typ
- コンパイル あり

### 3.4.1 手順

1. ターミナルで Typst をインストール  
`brew install typst`
2. path を通す
3. 拡張機能 Tinymist Typst をインストール
4. プレビューとコンパイルボタンが右上にで  
きるのでそこから実行

## 4 実例

同じようなドキュメントを作成した際の細かい違いと文法の違いについて示す。

### 4.1 MarkDown でレポート

#### 4.1.1 ソースコード

```
# MarkDownでレポート

- [MarkDownでレポート] (#markdownでレポート)
  - [数式] (#数式)
    - [ブロック] (#ブロック)
    - [インライン] (#インライン)

## 数式

### ブロック

ガウス積分：
$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

オイラーの公式：
$$e^{i\theta} = \cos \theta + i \sin \theta$$

自然対数の定義：
$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$$

テイラー展開：
$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

インライン

二次方程式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解は：

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$


判別式  $D = b^2 - 4ac$  により：


- $D > 0$  のとき、異なる2つの実数解
- $D = 0$  のとき、重解
- $D < 0$  のとき、共役な2つの複素数解

```

自然対数の定義：

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$$

テイラー展開：

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

### インライン

二次方程式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解は：

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

判別式  $D = b^2 - 4ac$  により：

- $D > 0$  のとき、異なる2つの実数解
- $D = 0$  のとき、重解
- $D < 0$  のとき、共役な2つの複素数解

### 4.1.2 PDF 出力

markdown.md 2026-01-08

#### MarkDownでレポート

- MarkDownでレポート
  - 数式
    - ブロック
    - インライン

数式

ブロック

ガウス積分：

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

オイラーの公式：

$$e^{i\theta} = \cos \theta + i \sin \theta$$

自然対数の定義：

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$$

テイラー展開：

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

インライン

二次方程式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解は：

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

判別式  $D = b^2 - 4ac$  により：

- $D > 0$  のとき、異なる2つの実数解
- $D = 0$  のとき、重解
- $D < 0$  のとき、共役な2つの複素数解

1 / 1

図 1: MarkDown

## 4.2 Typst でレポート

### 4.2.1 ソースコード

```
#import "@preview/js:0.1.3": *
#show: js.with(
    lang: "ja",
    seriffont-cjk: "Harano Aji Mincho",
    sansfont: "Harano Aji Gothic",
    sansfont-cjk: "Harano Aji Gothic"
)
#set page(
    paper: "a4",
    numbering: "1",
    columns: 1,
)
#align(center)[
    #text(
        size: 20pt,
    )[ Typstでレポート
    ]
    #v(2em)
    #text(
        size: 12pt,
    )[ 山根義疏 \
    #v(1em)
    #datetime.today().display("[year]年
[month]月[day]日")
    ]
]
#outline(
    title: "目次",
    depth: 2,
    indent: auto,
)
= 数式
== ブロック
ガウス積分：
$ integral_(-infinity)^(infinity) e^(-x^2)
d x = sqrt(pi)
$

オイラーの公式：
$ e^(i theta) = cos theta + i sin theta
```

```
$ 自然対数の定義：
$ lim_(n -> infinity) (1 + 1/n)^n = e
$

テイラー展開：
$ f(x) = sum_(n=0)^(infinity) frac(f^(n)(a), n!) (x-a)^n
$

== インライン
二次方程式 $a x^2 + b x + c = 0$ ($a != 0$)
の解は：
$ x = frac(-b plus.minus sqrt(b^2 - 4 a c), 2 a)
$

判別式 $D = b^2 - 4 a c$ により：
- $D > 0$ のとき、異なる2つの実数解
- $D = 0$ のとき、重解
- $D < 0$ のとき、共役な2つの複素数解
```

### 4.2.2 PDF 出力

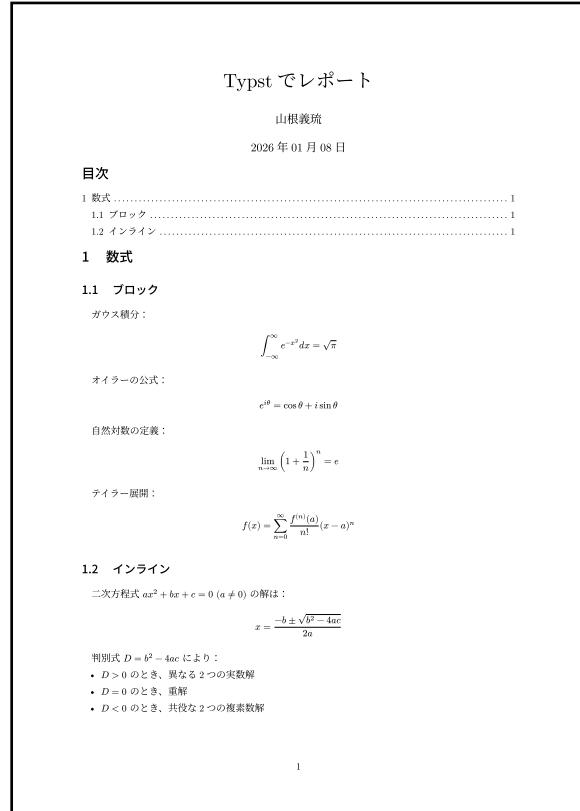


図 2: Typst

## 4.3 LaTeX でレポート

### 4.3.1 ソースコード

```
\documentclass{jlreq}
\usepackage[top=20truemm,bottom=20truemm,
, left=20truemm,right=20truemm]{geometry}

%
\title{\LaTeX でレポート}
\author{山根義琉}
\date{\today}
\begin{document}
\maketitle
\tableofcontents
\section{数式}
\subsection{ブロック}
ガウス積分：
\[
\int_{-\infty}^{\infty} e^{-x^2} dx =
\sqrt{\pi}
\]

オイラーの公式：
\[
e^{i\theta} = \cos\theta + i\sin\theta
\]

自然対数の定義：
\[
\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e
\]

テイラー展開：
\[
f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n
\]

二次方程式  $ax^2 + bx + c = 0$  ( $a \neq 0$ ) の解は：
の解は：

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

判別式  $D = b^2 - 4ac$  により：
\begin{itemize}
\item  $D > 0$  のとき、異なる2つの実数解
\item  $D = 0$  のとき、重解
\item  $D < 0$  のとき、共役な2つの複素数解
\end{itemize}
\end{document}
```

```
\item  $D > 0$  のとき、異なる2つの実数解
\item  $D = 0$  のとき、重解
\item  $D < 0$  のとき、共役な2つの複素数解
\end{itemize}
\end{document}
```

### 4.3.2 PDF 出力

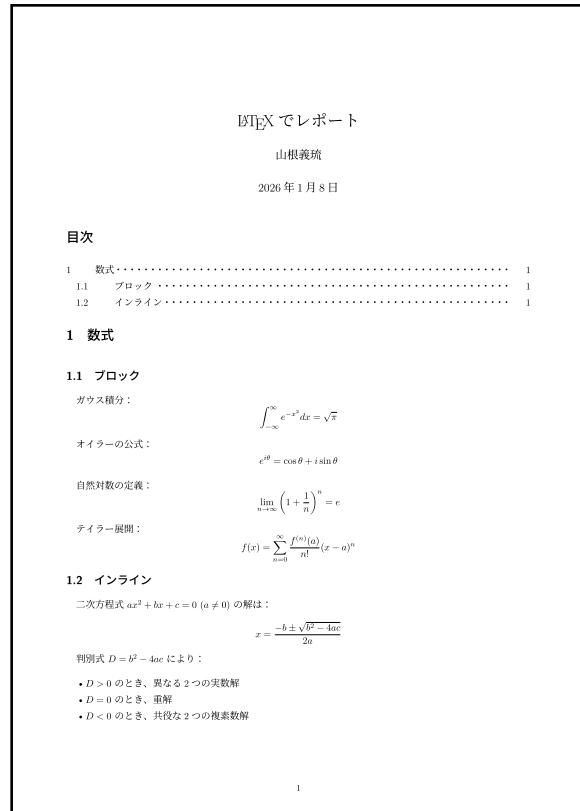


図 3: LaTeX