A Practitioner's Guide to basic LaTeX Setup

Contents

A Practitioner's Guide to Utilization of TEX Engine[†]

There are various commercial and open source word processors such as Microsoft Word, Writer[1], and Pages[2]. These alternatives all belong to the same category: WYSIWYG[3] editor. Text formatting is the essential demanding function for modern computers. In this note, we will introduce a typesetting system, XƎLATEX, and compare their advantages and disadvantages towards current existing word processors.

1 Introduction

TEX is a typesetting system developed by Donald Knuth, a Computer Science Professor at Stanford University in 1978. After the release of this language, a myriad of engines have been introduced including LATEX, XƎLATEX, and LuaLATEX. These engines are set of macros that help us program the TEX language.

The engine that prevails across user selection belongs to LATEX. However, our focus, XƎLATEX is another modification of the underlying TEX engine, very similar to LATEX from user's perspective and can be treated as a side branch of it. The advantage of XƎLATEX is the ability to use system fonts and customized fonts including Chinese[4] and Japanese, rather than limited English font package in LATEX but the draw back is the performance is slightly slower.

Before we begin digging into the details of XƎLATEX, we will provide a comparison between using Microsoft Word and LATEX(set as comparison base).

- Disadvantages
    1. *Accessibility*: Steep learning curve. Does not compile if there is any coding error.
    2. *Efficiency*: Editing is slower than WYSIWYG editors.
    3. *Popularity*: Word has a much higher market share in industry and across households.
- Advantages
    1. *Availability*: Price free. Language(TEX) is bug free which makes LATEX very stable.
    2. *Structuralization*: Elegant math type. Separate the content from the format (style) of the document. Code commenting is available.
    3. *Compatibility*: Cross platform, all versions are compatible. Always produce exact same documents across time(years). More expandable and customizable.
    4. *Compactness*: Produce small and lean documents (Word causes trouble with large documents). Auto tracking and leading optimization.

[†] This note is written by Hao-Che Hsu with XƎLATEX and serve as a supplementary material for introduction to TEX typesetting system. All errors and omissions belong to the author. Comments Welcome. This manuscript may be printed and reproduced for individual or instructional use, but may not be printed for commercial purposes.

[1] Writer is a component of OpenOffice (project discontinued in 2011), an open source office suite: https://www.openoffice.org or LibreOffice: https://www.libreoffice.org.

[2] Pages is developed by Apple. Previously, it is part of the iWork productivity suite and runs on macOS.

[3] WYSIWYG (WIZ-ee-wig) is an acronym for “what you see is what you get” (Wikipedia).

[4] Utilizing XƎLATEX enables us to bypass the usage and setup of cwTEX (default provide 5 Chinese fonts only).

5. *Security*: No UUID[5]. No macro viruses. TEX documents received by email is anti-viruses.

LATEX (lay-tech) is widely used in scientific areas and academia so there are abundant notes and tutorials available on the Internet and forums. However, most of them are very detailed into a large number of topics which makes them hard to follow.

This note intends to cover only those useful functions that readers are highly likely to encounter in school work and academia. Readers will find this note easy to follow, practical with step by step guiding and really helpful to produce XƎLATEX (zee-lay-tech) documents.

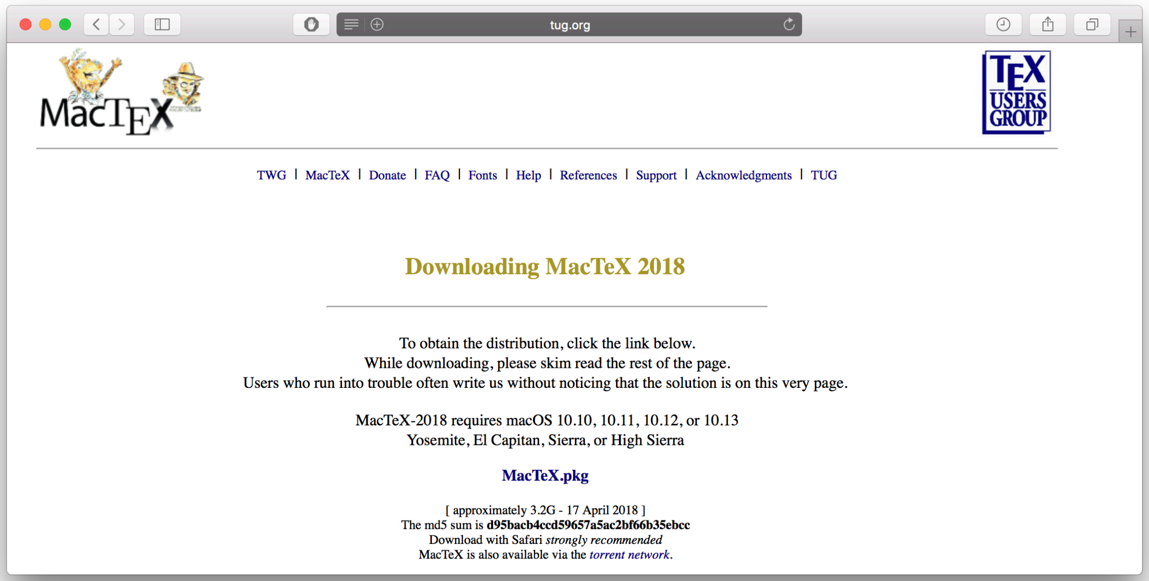
2 Installation

The environment settings involves two steps: language installation and editor installation. There are also online editor such as ShareLaTeX, Overleaf, and Authorea, but in this note we recommend local, offline working environment.

2.1 Install TEX Language

If you are a Mac[6] user, we will do a one-time installation of the language from MacTEX distribution: http://www.tug.org/mactex/mactex-download.html. MacTEX is a redistribution of TEXLive[7]. This full installation will take space up to 4GB and contains 3 subpackages.

Figure 1: MacTEX Download website


tug.org
MacTEX
TEX USERS GROUP
TWG | MacTeX | Donate | FAQ | Fonts | Help | References | Support | Acknowledgments | TUG

Downloading MacTeX 2018

To obtain the distribution, click the link below.
While downloading, please skim read the rest of the page.
Users who run into trouble often write us without noticing that the solution is on this very page.

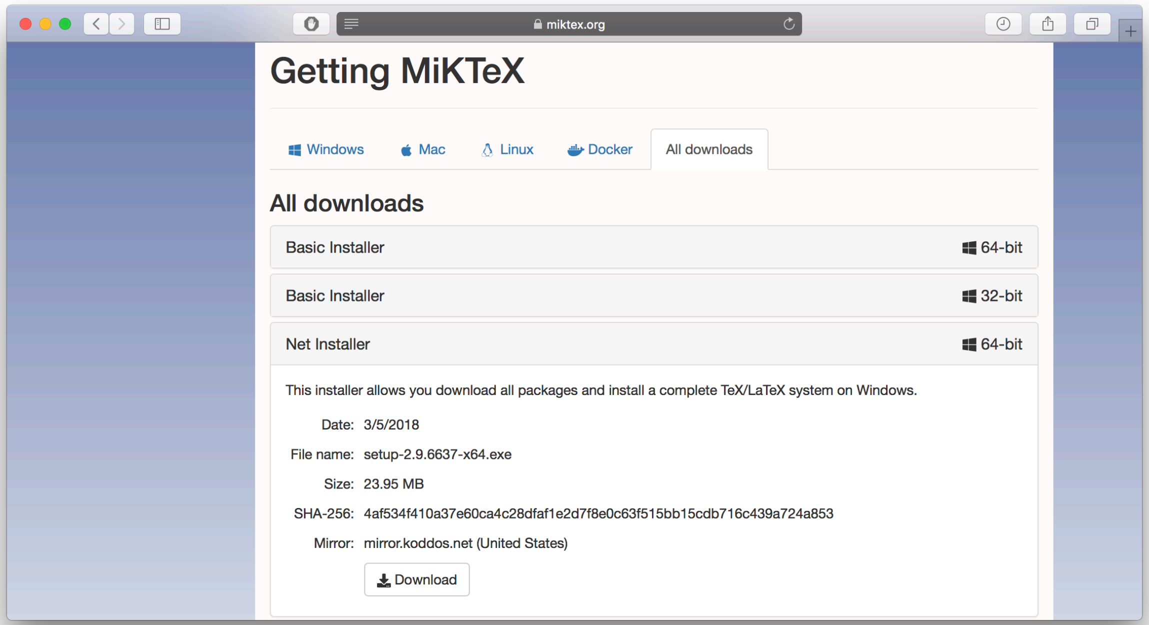
MacTeX-2018 requires macOS 10.10, 10.11, 10.12, or 10.13
Yosemite, El Capitan, Sierra, or High Sierra

MacTeX.pkg

[approximately 3.2G - 17 April 2018]
The md5 sum is **d95bacb4ccd59657a5ac2bf66b35ebcc**
Download with Safari *strongly recommended*
MacTeX is also available via the *torrent network*.

[5] A universally unique identifier (UUID) is a 128-bit number used to identify information in computer systems (Wikipedia). The UUID in Word documents embed a code that can trace back to our computer.
[6] The Macintosh system refers to macOS.
[7] TeX Live is a free distribution for the TeX typesetting system, up to version 2009 it could be run directly from a CD-ROM (Wikipeda).

For Windows user, we will install from MiKTEX distribution: https://miktex.org/download.



Getting MiKTeX

All downloads

Basic Installer	64-bit
Basic Installer	32-bit
Net Installer	64-bit

This installer allows you download all packages and install a complete TeX/LaTeX system on Windows.

Date: 3/5/2018
File name: setup-2.9.6637-x64.exe
Size: 23.95 MB
SHA-256: 4af534f410a37e60ca4c28dfaf1e2d7f8e0c63f515bb15cdb716c439a724a853
Mirror: mirror.koddos.net (United States)

Do not select the basic installer but choosing the "Net Installer"[8] instead. Despite the installer is few MBs only, the full package is several gigs which requires hours to acquire all the files.

After the download is completed we will need to find the `setup-version-x64.exe`[9] with the above icon among the 3,500 files to initiate the install.

2.2 The LATEX Widget

For Mac user, after you install from `MacTeX.pkg` there will be a centralized `TeX` folder in your `Applications` folder. There will be some specification documents, utilities and widgets which you might find it useful.

`TeXShop` is the basic macOS platform TEX editor. But this app is not recommended since it only includes the fundamental functions.

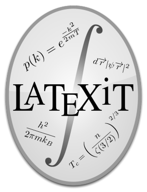
On the other hand the widget `LaTeXiT` can be really useful. There are several math editors available including the `Equation Editor` in Word and the famous commercial package `MathType`.

[8] The net installer is a *downloader* which gathers the essential files before installation. But this is a decentralized downloading process, so user requires to direct the download destination to a specific folder.

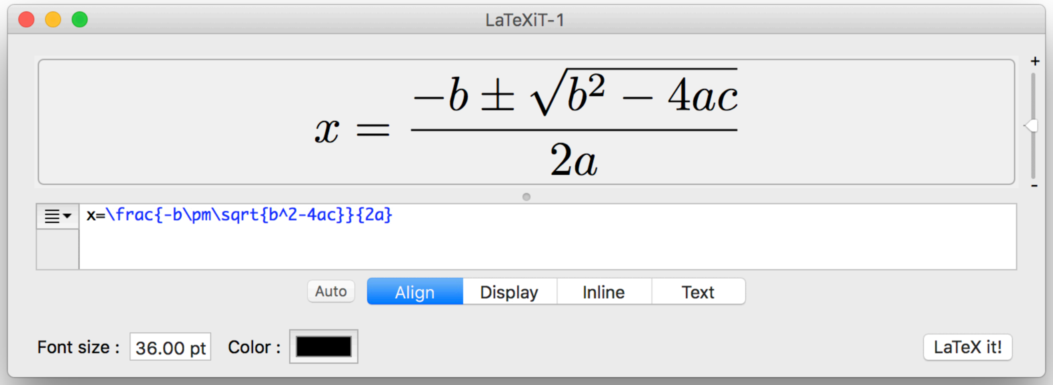
[9] To determine our Windows installed version, user can go to **control panel** → **System and Security** → **System** to check our system type (32-bit or 64-bit).

Figure 2: TeXShop

Figure 3: LaTeXiT



However, MathType does not fully compatible with the latest Microsoft Office suite and iWork suite[10] on macOS platform. After you are familiar with LATEX syntax, you will be able to manipulate LaTeXiT as a substitute to any other equation editors.



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

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

After type the equation syntax[11] into the console and click the LaTeX it! button, the widget will generate the visualized equation in several choosable format[12] and we can directly drag the object into PowerPoint, OmniGraffle, Pages or any other desired visual editors. To edit the equation, double click the inserted equation and the LaTeXit console will reappear. User can then fix the equation and recompile, the updated equation will change automatically at the dragged destination.

2.3 The LATEX Editor

After installing the TEX language, we need to choose our own favorite editor. There are both commercial and open source editors available. Most of them support multi-platform. We will list some of them for user to choose based on their preference:

- TeXworks: open source, based on TeXShop, https://www.tug.org/texworks/.
- Lyx: free, WYSIWYM[13] editor, https://www.lyx.org.
- TeXstudio: open source, a branch of TeXmaker, more customizable, https://www.texstudio.org.

[10]Currently, the name iWork is not primarily used. The suite is now released as separate apps: Pages, Number, Keynote.
[11] Learning the syntax is the crucial step to be a successful LATEX user. In this example, the used equation syntax are the following: fraction(\frac{·}{·}), plus-minus(\pm), square root(\sqrt{·}), power script(∧).
[12] The supported format includes: PDF, EPS, TIFF, PNG, JPEG.
[13] WYSIWYM is an acronym for "what you see is what you mean" (Wikipedia).

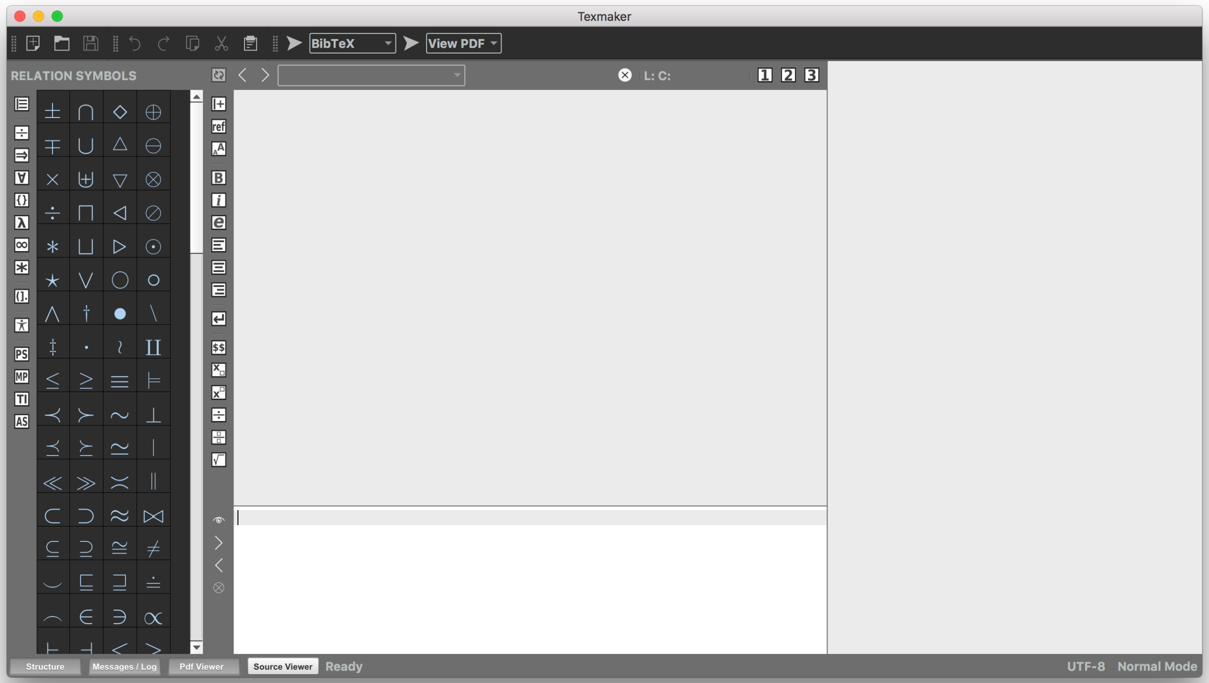
- TeXmaker: open source, http://www.xm1math.net/texmaker/.
- WinEdt: shareware, only support Windows, http://www.winedt.com.

Other then these integrated environment, users can also choose pure editor with extensions support to serve as ideal LATEX editor. These editor are powerful but may require more experienced user.

- Sublime Text: commercial package.
- Emacs: free.
- Atom: free, open source.
- Visual Studio Code: free.
- Vim(with LATEX-suite): text editor program for Unix.
- TeXlipse: free, a plugin that adds Latex support to the Eclipse IDE[14].
- Gummi: An alternative for Linux[15] (e.g. Ubuntu).

In this note, we recommend `Texmaker`, a powerful app with beautiful interface that can be really helpful when writing LATEX code.

Figure 4: Texmaker interface



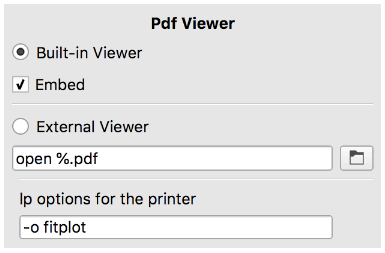
[14] Eclipse is a very powerful IDE that supports object oriented programming to develop Java, C++, Python and in this case, LATEX. Its native is Java development. Plugin is required for other languages.
[15] Gummi requires to install from source code via Linux development tools.

3 Texmaker Basic Settings

We will need to make some basic adjustments, tuning the settings to best fit user accommodation. The settings are located in `preference` tab.

3.1 Commands

In the `commands` tab, we will check the `Embed` box. This will position the pdf viewing screen besides (on the right) the editor. In this case, we can view the compiled document and edit the contents at the same time.



Pdf Viewer

[● Built-in Viewer]

[✓ Embed]

○ External Viewer

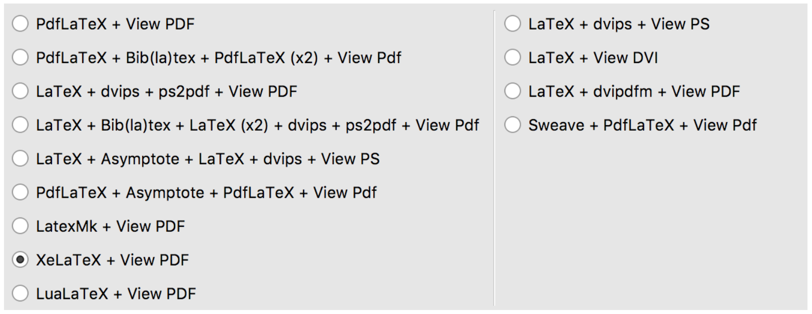
open %.pdf

lp options for the printer

-o fitplot

3.2 Quick Build

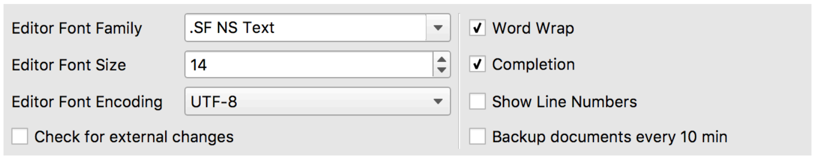
The default *Quick Build* (hot key: F1) is *PdfLaTeX + View PDF*. This is also the most common compiling setting. Most CV, résumé, cover letter templates found on the Internet requires this LATEX engine. However, we need to choose XƎLATEX to take advantage of its font customization[16].



○ PdfLaTeX + View PDF
○ PdfLaTeX + Bib(la)tex + PdfLaTeX (x2) + View Pdf
○ LaTeX + dvips + ps2pdf + View PDF
○ LaTeX + Bib(la)tex + LaTeX (x2) + dvips + ps2pdf + View Pdf
○ LaTeX + Asymptote + LaTeX + dvips + View PS
○ PdfLaTeX + Asymptote + PdfLaTeX + View Pdf
○ LatexMk + View PDF
[● XeLaTeX + View PDF]
○ LuaLaTeX + View PDF
○ LaTeX + dvips + View PS
○ LaTeX + View DVI
○ LaTeX + dvipdfm + View PDF
○ Sweave + PdfLaTeX + View Pdf

3.3 Editor

Finally, user can customize font and size to make the editor more readable. Here we recommend to open word warp and close line numbers to maximize the space of viewing screen.



Editor Font Family	.SF NS Text
Editor Font Size	14
Editor Font Encoding	UTF-8

☐ Check for external changes

[✓ Word Wrap]

[✓ Completion]

☐ Show Line Numbers

☐ Backup documents every 10 min

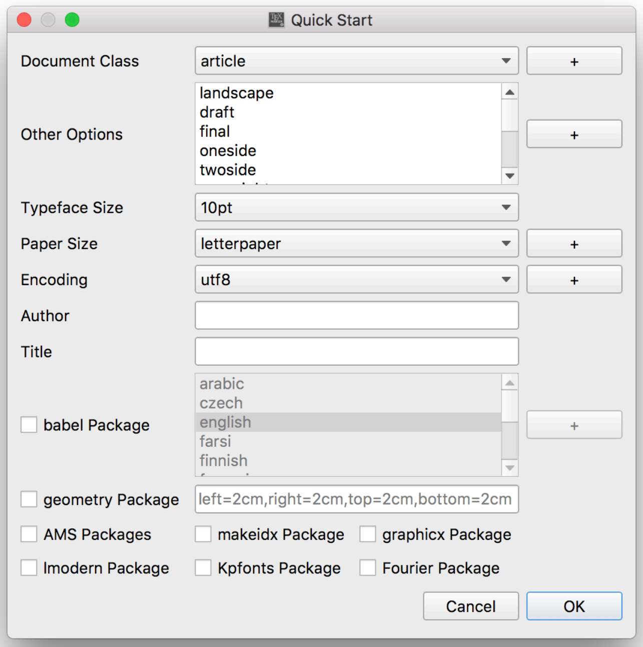
[16] Without changing font, these two compiling settings are the same.

4 The Coding

This section will serve as a tutorial to help you write LATEX code from scratch. Writing LATEX code is analogous to Python. You will need to import a lot of packages (as modules in Python) and make your own customization. Beware that a single error will make your code end up not compile, so incremental coding[17] is recommended.

4.1 Quick Start

When you initiate Texmaker with a new script, the first thing is always to `save` to a **folder** first before typing anything. To construct the basic skeleton code, we can go to the upper menu and select `Wizard` → `Quick Start`:



Quick Start

Document Class	article	+
Other Options	landscape draft final oneside twoside	+
Typeface Size	10pt	
Paper Size	letterpaper	+
Encoding	utf8	+
Author		
Title		
☐ babel Package	arabic czech english farsi finnish	+
☐ geometry Package	left=2cm,right=2cm,top=2cm,bottom=2cm	
☐ AMS Packages	☐ makeidx Package	☐ graphicx Package
☐ lmodern Package	☐ Kpfonts Package	☐ Fourier Package

Cancel OK

The document class is `article` with font `10pt` and a letter-size paper. The document will need to encode with `UTF-8`. Then we will obtain the following default code:

```
\documentclass[10pt,letterpaper]{article}
\usepackage[utf8]{inputenc}
   ↕import packages, settings, macros
\begin{document}
   ↕contents
\end{document}
```

17 Incremental coding ensures you to find bugs(errors) as soon as possible. You should not compile everything at the end but compile your code when you finish every little pieces in order to trace the error if there is any.

As indicated by purple at the bottom of the previous page, all the imported packages need to be declared before the *document* environment:

`\usepackage{first package name, second package name, third,...}`

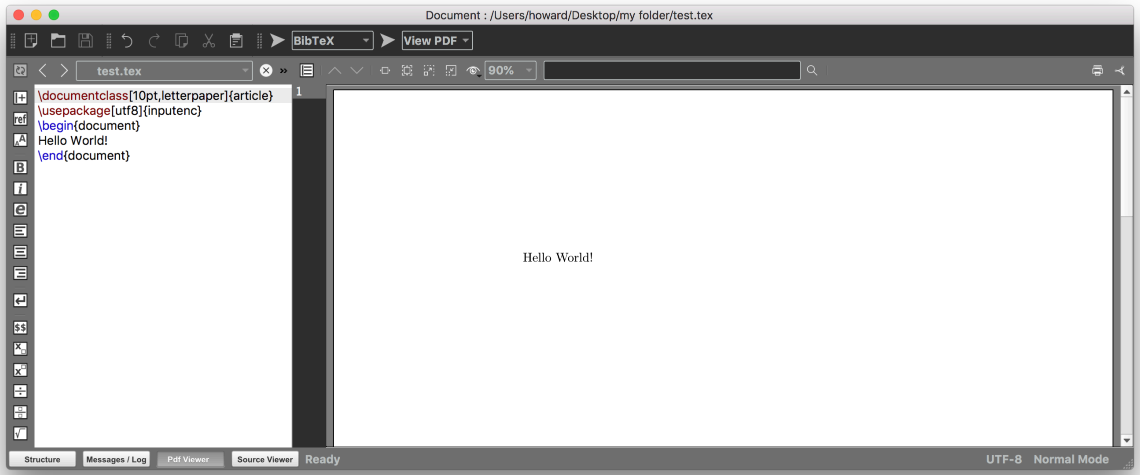
where an *environment*, always a pair of clauses, is specified between `\begin` and `\end`. The *name* of the environment is specified right after the clause:

`\begin{environment name}` ··· `\end{environment name}`

And all the contents need to be put in the *document* environment. Try typing "Hello World!" inside the document environment:

```
\documentclass[10pt,letterpaper]{article}
\usepackage[utf8]{inputenc}
\begin{document}
Hello World!
\end{document}
```

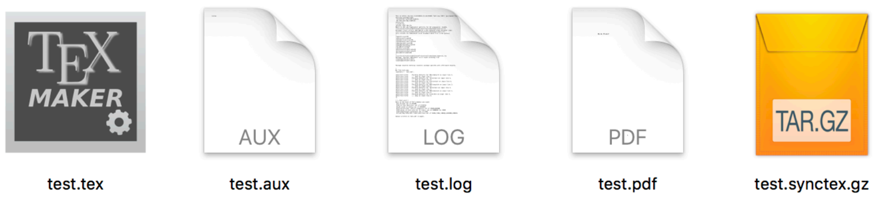
Then go to the menu bar: `Tools` → `Quick Build` or press `F1`. Then a PDF file will be created.


Document : /Users/howard/Desktop/my folder/test.tex

```
\documentclass[10pt,letterpaper]{article}
\usepackage[utf8]{inputenc}
\begin{document}
Hello World!
\end{document}
```

Hello World!

If you check the folder where you save this file, you will notice that texmaker has produce four[18] additional files. The `.pdf` is our document. The `.tex` is our LATEX script. The rest are log files which we can ignore them.



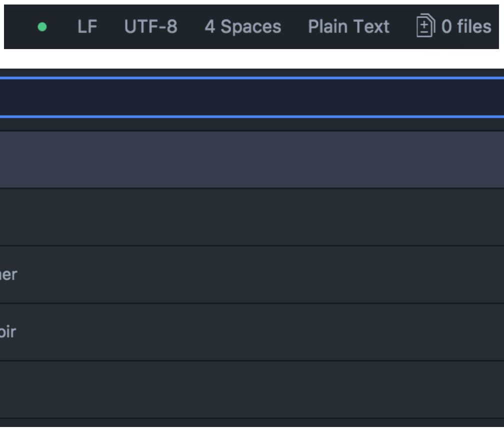
[18] Later when we learn bibliography (bibTEX), referencing, more log files will be produced when compiling.

4.2 Code Organizations

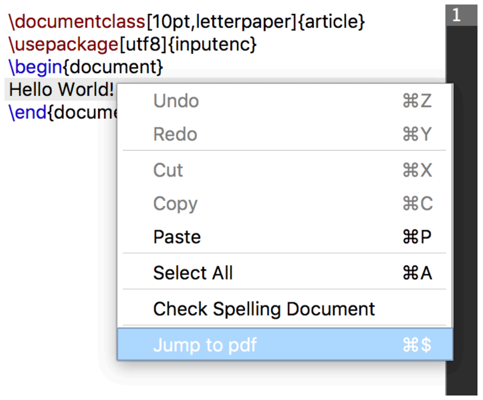
When we are writing a book, a report or a thesis, it is likely that we will have thousands of lines of codes. The codes not only will involve contents but also math formulas, tables, and different environments.

There are two ways to sort out the mess. First, when we code, it is a good habit to indent our code. But unlike Java or Python, when we write LATEX code we wish to focus *only* on our contents and let TEX handle the rest of the settings e.g. page, line height adjustments. Thus, we recommend an open source editor, Atom[19]: https://atom.io.

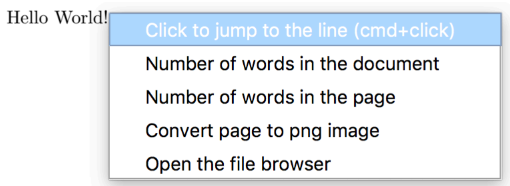
We can copy our code to Atom, go to the bottom right and change `Plain Text` to `LaTeX`.



Then we can go to Atom's menu bar: `Edit` → `Lines` → `Auto Indent`[20]. Then our LATEX code will be correctly indented which we can paste it back to Texmaker. There is an alternative approach.



```
\documentclass[10pt,letterpaper]{article}
\usepackage[utf8]{inputenc}
\begin{document}
Hello World!
\end{docume
```


Hello World!
Click to jump to the line (cmd+click)
Number of words in the document
Number of words in the page
Convert page to png image
Open the file browser

[19] Atom is an open source, highly customizable (fully "hackable") cross platform editor developed by Github.
[20] Before using auto-indent, make sure there is no *trailing whitespace* (extra spaces) at the end of *each* line.

We can directly right click the targeting code and select `jump to pdf` (figure on the bottom left of previous page), the viewer will then jump to the corresponding location. In the same way, we can right click on the viewer to locate the corresponding code in the editor (right figure).

4.3 Page Organization

The first essential package we want to introduce is the *geometry* package. Please copy the following code and put it before the document environment in a **single line**:

```
\usepackage[letterpaper,left=25mm,right=25mm,top=25mm,bottom=25mm,bindingoffset=0mm]
{geometry}
```

For simplicity, we will call the *document environment*: **main**. This follows the programming convention in Java that all the executing content must be in the main method[21]. Using this concept, **before main** indicates the space before `\begin{document}` (first vertical arrow at the bottom of page 8) and **in main** means the space between `\begin{document}` and `\end{document}` (second purple arrow). Hence, after adding the geometry package before main, our code will look like the following (with word warp[22]):

```
1 \documentclass[10pt,letterpaper]{article}
2 \usepackage[utf8]{inputenc}
3 \usepackage[letterpaper,left=25mm,right=25mm,top=25mm,bottom=25mm,bindingoffset=0mm]{geo
4 metry}
5 \begin{document}
6 Hello World!
7 \end{document}
```

User can adjust the length from the edge of the paper from 4 sides and switch the unit at will (e.g. change to cm). Here we recommend 25 millimeters(mm). The binding offsets determines the horizontal shifts to leave spaces for the staples. The first argument, `letterpaper`, in line 3 indicates the paper size. This argument needs to match with the second argument in the first line. For international users, we can change the argument `letterpaper` to `a4paper`.

4.4 Header and Footer

The header and footer refer to six different positions. Three at the top and the other three at the bottom. The package we are going to use is `fancyhdr`.

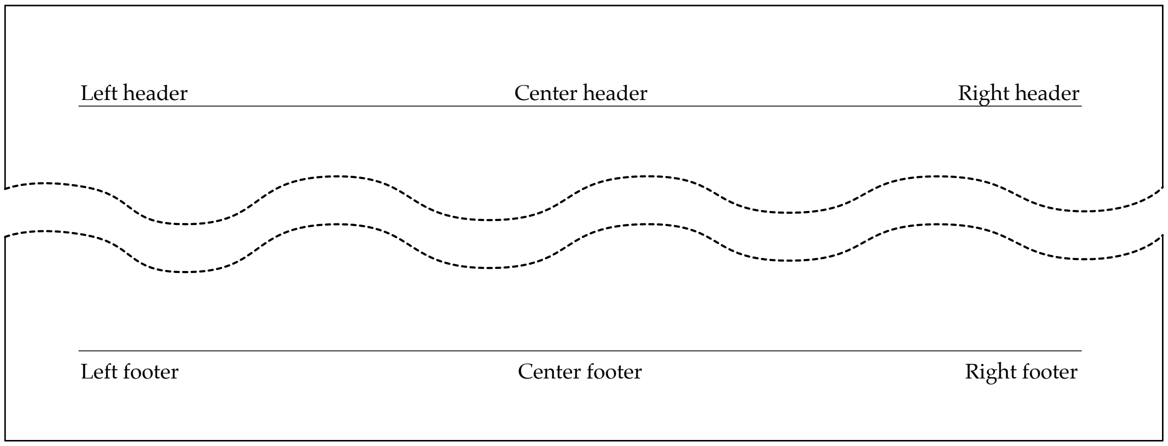
This package gives you full customization of the six positions. You can draw lines, change fonts, specify different style for different pages and set up global style for every even and odd pages. To enable `fancyhdr`, we need to import the package, i.e. add the following code before main:

```
\usepackage{fancyhdr}
```

[21] The execution of a Java program reads a sequence of Java code (instructions) in a certain order. To define a starting point, a Java program starts by executing the main method (static method) of some class. This is analogous to LaTeX coding as the contents are all in the document environment.

[22] *Word wrap* unlike *word break* is a feature in text editors and word processors that automatically moves the characters to the next line when reaching the end of the current line without requiring you to press enter (Computer Hope).

Below is a sketch of a page. Looking at the top and bottom are six positions with 25mm to the four sides.



This style is realized by the settings in `fancyhdr`. Following is the code which customizes the contents of the six positions. We will need to put the code after `\usepackage{·}` and before main.

```
1  \fancypagestyle{style name}{
2    \fancyhf{
3    }
4    \fancyhead[C]{Center header}
5    \fancyhead[R]{Right header}
6    \fancyhead[L]{Left header}
7
8    \fancyfoot[C]{Center footer}
9    \fancyfoot[R]{right footer}
10   \fancyfoot[L]{left footer}
11
12   \renewcommand{\headrulewidth}{0.4pt} %Draw header separation line (thickness 0.4pt)
13   \renewcommand{\footrulewidth}{0.4pt} %Draw footer separation line
14 }
```

Before we go to the variation settings of `fancyhdr`, first we need to review this code. The settings for this package is defined by `\fancypagestyle{·}{·}`. The first clause is the tag, a name you choose for this setting. The second clause is the customization which opens at line 1 and closes at line 14 in our example above. To make a clean definition, we will erase all the existing settings by making the header and footer empty: `\fancyhf{}`[23], i.e. leaving the brackets empty, just as line 2 above. Then we need to specify the *position* and *layout* indicator:

Position Indicator:

- C: center
- R: right
- L: left

Layout Indicator:

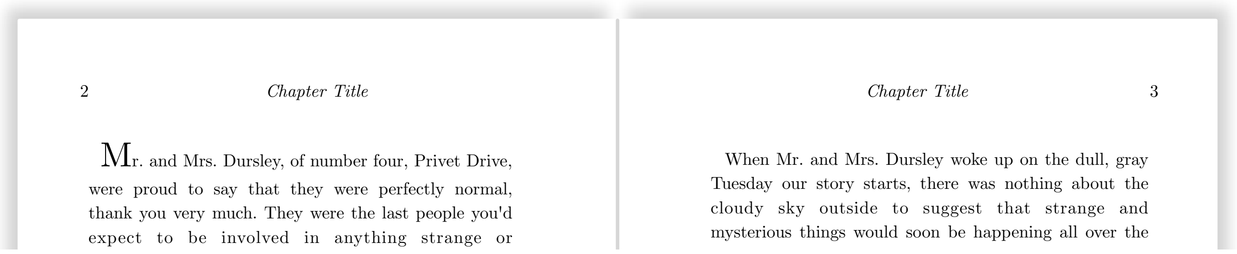
- E: even page
- O: odd page

[23] This command sets "facy header and footer "at the same time.

Following the code in line 4-6 and 8-10, we can add contents to our six head and footer positions by:

$$\backslash\text{fancy}\begin{Bmatrix}\text{head}\\\text{foot}\end{Bmatrix}\text{[indicator]\{contents\}}$$

Notice that if we are editing a *book*, the layout is going to be quit different to an *article*:



2 *Chapter Title*

Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or

Chapter Title 3

When Mr. and Mrs. Dursley woke up on the dull, gray Tuesday our story starts, there was nothing about the cloudy sky outside to suggest that strange and mysterious things would soon be happening all over the

In above example, we have the chapter title at the header center and for the even page, the page number is at the left corner. For the odd page, the number is at the right corner. The specification requires double indicator:

$$\backslash\text{fancyhead[EL]\{}\backslash\text{thepage\}} \tag{1}$$

$$\backslash\text{fancyhead[OR]\{}\backslash\text{thepage\}} \tag{2}$$

and only works under the "book"environment. Recall in 4.1, we define our document class: *article* (first line). To change to *book* class, we just need to make a small adjustment to the first line of the code:

$$\backslash\text{documentclass[10pt,letterpaper]\{book\}}$$

which is changing *article* to *book*. Under book class, double indicator will work. Now back to our (1) code above, "EL"stands for "Even-page-Left"which makes all the page number shown on the left header for all even pages. Analogously, "OR"will display the page number at the right header for all odd pages.