目录

书籍导出

编辑工具

发布书籍

结束语录

简单介绍

`Gitbook` 是什么？其实用一句话就可以概括，它是一个 **能将使用 Markown 语法的 `md` 格式文档，快速制作成各种格式电子书的工具**。

常被用于编写文档或者电子书，特点是方便简洁，易于使用。只要熟悉轻量级标记语法的 `Markdown` 语法，就能使用Gitbook 来制作各种格式的电子书。

笔者写了这本电子书，详细的分享一下自己使用 Gitbook 的相关方法和经验，欢迎大家的浏览和分享，本书遵循 `MIT` 协议许可，**转载需要注明来源**。

相关概述

1. 看完上面的介绍，是不是知道了 Gitbook 是什么，主要用于做什么了？其实就是将我们的书写的文章生成对应格式的电子书，方便分享大家浏览，比如生成pdf给其他人查看相关的内容，或者生成静态 HTML，发布到网站服务器，可以通过在线电子书的方式分享给更多的人。
2. 与 Git 工具对应的 Github 仓库一样，Gitbook 也有一个官方的仓库（可以在线编辑制作），可以上传我们的电子书的文件，不过在国内貌似部分用户访问不了，但是也没有关系，因为大多数情况主要还是利用这个工具生成的对应的格式电子书进行分享或者发布。
3. 总之，**Gitbook 就是一个电子书生成工具** ，类似与 Git ， `Git` 是一个代码仓库管理工具，用于管理代码文件，并且可以生成代码的变更记录，同时具备上传这些文件和变更记录到指定的服务器。那么同理，我们也可以结合 Gitbook 和 Git 来管理我们的文档和生成的电子书文件。当然，本书主要介绍 Gitbook，关于Git 的相关知识，可以参考其他相关的教程。

本书概况

本书将分 `简单介绍` 、 `安装使用` 、 `结构配置` 、 `插件使用` 、 `书籍导出` 、 `编辑工具` 、 `发布书籍` 、 `结束语录` 八个部分，分别来介绍 Gitbook 相关的基础知识和相关使用方法和笔者的一些结束语以及联系方式。

- 简单介绍：对 Gitbook 进行简单的介绍了解，本书相关概况信息以及相关网站参考推荐。

- 安装使用：介绍如何安装 Gitbook 环境，和一些基本的命令的使用。
- 结构配置：介绍如何 Gitbook 项目的目录结构，如何进行相关的配置和使用。
- 插件使用：介绍了如何配置和使用插件，并介绍了一些第三方的插件和相关的使用方法。
- 书籍导出：介绍了如何将书籍编译和导出各种类型的电子书文件。
- 编辑工具：介绍和列举了一些相关的 Mardown 文档的可视化编辑器工具。
- 发布书籍：介绍了几种常见的方式将自己创作的书籍发布分享出去，供大家浏览。
- 结束语录：主要对本书的做了一个简单的总结，碎碎念，回顾及相关联系方式。

相关网站参考

1. Gitbook 项目官网：https://www.gitbook.com
2. Gitbook 项目GitHub：https://github.com/GitbookIO/gitbook
3. Npm Toolchain：https://www.npmjs.com/
4. ebook-concert：https://calibre-ebook.com
5. JSON 语法参考：https://www.json.org/json-zh.html
6. Markdown 语法参考：http://markdown.p2hp.com
7. Git 官网：https://git-scm.com

MIT许可

```
Copyright 2021 ©JiangMing all right reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of tl

The above copyright notice and this permission notice shall be included in all cop

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIEI
```

1. 安装介绍

这一章主要介绍本地 `Gitbook` 相关环境的安装，目前主流的平台：Windows、Linux、Unix，Mac OS都是支持的。本地安装需要依赖 `Node.js` 环境 ，如果需要输出 PDF， eBook，Mobi 等格式的电子书还需要安装 `ebook-concert` 依赖，好了，下面进入正题!

2. 环境准备

2.1.1. 下载和安装 Node.js

- 官网：https://nodejs.org/
- 官网（中文）：https://nodejs.org/zh-cn
- 中文镜像网站：http://nodejs.cn
- 淘宝的镜像下载：https://npm.taobao.org/mirrors/node

进入上面网站进入选择相关平台下载对应的版本的 Node.js 方法或者安装包，一种是官网根据不同的平台对应的命令来安装，还有一种是下载二进制安装包（笔者推荐方法），进行环境变量配置，本文将介绍笔者的环境安装 Node.js，笔者使用的是 `Linux Ubuntu` 环境，使用二进制安装包来安装。

> 注意: 基于截止到目前的 Gitbook V3.2.3版本，需要使用NodeJs的v10+版本，否则会产生各种报错。 推荐使用：node-v10.24.0-linux-x64

下载 node-v10.24.0-linux-x64 版本的压缩包后，解压到特定的目录，进行环境变量配置。

1. 插入logo

```
执行 vim ~/.profile 或者 vim ~/.bashrc 后添加下面的环境变量配置

$ vim ~/.profile
export DART_HOME=TargetPath/node-v10.24.0-linux-x64
export PATH=${DART_HOME}/bin:$PATH
```

安装完成之后，可以通过下面的命令来验证一下 `Node.js` 是否安装成功。

```
$ node -v
v10.24.0
```

2.1.2. 安装 ebook-concert 依赖

`ebook-concert` 主要用于生成 PDF、eBook，Mobi 等格式的电子书，具体各平台的下载安装参考官网：https://calibre-ebook.com/download

这里我是使用的Ubuntu平台，执行下面的命令即可：

```
sudo -v && wget -nv -O- https://download.calibre-ebook.com/linux-installer.sh |  su
```

安装完成之后，可以通过下面的命令来验证一下 `ebook-concert` 是否安装成功。

```
$ ebook-convert --version
ebook-convert (calibre 5.13.0)
Created by: Kovid Goyal <kovid@kovidgoyal.net>
```

3. 安装 gitbook-cli

执行安装命令, 可参考：https://www.npmjs.com/package/gitbook

```
$ npm config set registry http://registry.npm.taobao.org   // 可选，设置一下淘宝镜像
$ npm install gitbook-cli -g
```

安装完成之后，可以通过下面的命令来验证一下 `Gitbook` 是否安装成功。

1. 插入logo

```
$ gitbook -V
CLI version: 2.3.2
GitBook version: 3.2.3
```

看到正确的输出版本信息，到此就完成了 `Ubuntu` 平台的 `Gitbook` 的安装了。

1. Gitbook命令行的使用

`Gitbook` 的命令行工具其实就是 `gitbook-cli` ，可以通过命令的方式来创建，构造，安装插件，预览等功能。

2. 查看帮助

`Gitbook` 和 `Git` 一样是一个命令行工具，开始介绍之前，先使用 `gitbook help` 命令来概览看一下 gitbook 主要的几个命令。

1. 插入logo

```
$ gitbook help
build [book] [output]       build a book
   --log                    Minimum log level to display (Default is info; Values a
   --format                 Format to build to (Default is website; Values are webs
   --[no-]timing            Print timing debug information (Default is false)

serve [book] [output]       serve the book as a website for testing
   --port                   Port for server to listen on (Default is 4000)
   --lrport                 Port for livereload server to listen on (Default is 35
   --[no-]watch             Enable file watcher and live reloading (Default is tru
   --[no-]live              Enable live reloading (Default is true)
   --[no-]open              Enable opening book in browser (Default is false)
   --browser                Specify browser for opening book (Default is )
   --log                    Minimum log level to display (Default is info; Values a
   --format                 Format to build to (Default is website; Values are webs

install [book]              install all plugins dependencies
   --log                    Minimum log level to display (Default is info; Values a

parse [book]                parse and print debug information about a book
   --log                    Minimum log level to display (Default is info; Values a

init [book]                 setup and create files for chapters
   --log                    Minimum log level to display (Default is info; Values a

pdf [book] [output]         build a book into an ebook file
   --log                    Minimum log level to display (Default is info; Values a

epub [book] [output]        build a book into an ebook file
   --log                    Minimum log level to display (Default is info; Values a

mobi [book] [output]        build a book into an ebook file
   --log                    Minimum log level to display (Default is info; Values a
```

1. 插入logo

> [!WARNING|style:flat]
>
> - 其中`[]`标记的是 **可选参数** 指令，但是有可能在某些平台中不生效甚至会报错。可能是 gitbook 在平台的兼容性上还不够完善原因导致的，如有参数使用的需求的情况，可以根据需要选择适合的平台来使用。目前笔者测试过在部分 `Ubuntu` 环境平台上部分参数指令可能无效（可能简写命令格式支持，或许支持完整参数格式又或许都不支持），但是在 `Windows` (windows 10) 平台上是可以使用的，具体的参数是否可正常的使用取决于不同平台的支持程度。
> - `--` 标记的是可选的功能参数，如 --log 输出执行相关的log，具体功能描述参考上面help中后面的描述。

简单的了解一下这些参数吧：

- [book]：指定 `gitbook` 项目的目录
- [output]：指定文件输出的目录

使用示例：

```
# 在指定的/home/gitbook/目录中初始化一个书籍项目
$ gitbook init --book=/home/gitbook/
# 对应的简写格式
$ gitbook init /home/gitbook/

# 指定书籍项目目录在当前目录，并将编译构建后文件放到指定的当前目录下的mybook目录中
$ gitbook build --book=./  --output=./mybook
# 对应的简写格式
$ gitbook build ./ ./mybook

# 指定书籍项目目录在当前目录，导出 PDF 格式的电子书到指定的/home/pdf/目录中
$ gitbook pdf --book=./  --output=/home/pdf/
# 对应的简写格式
$ gitbook pdf ./ /home/pdf/
```

3. 初始化

使用 `gitbook init` 初始化一本书，本地会默认创建生成两个 `markdown` 格式的文件，这两个文件是必须存在的，一个是初始化页面，一个是电子书的目录结构定义文件。

1. 插入logo

```
# 电子书初始化
$ gitbook init
warn: no summary file in this book
info: create README.md
info: create SUMMARY.md
info: initialization is finished

# 查看初始化的目录结构
$ tree
.
├── README.md
└── SUMMARY.md
```

4. 构建电子书

使用 `git build` 命令来生成静态网页格式的电子书。执行后会生成HTML静态资源输出到当前项目的目录下 `_book` 目录中。

1. 插入logo

```
# 构建电子书
$ gitbook build
info: 21 plugins are installed
info: 18 explicitly listed
info: loading plugin "highlight"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "code"... OK
info: loading plugin "splitter"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "github"... OK
info: loading plugin "alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "popup"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "donate"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 3 pages
info: found 0 asset files
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 0.7s !

# 查看简单的目录结构
$ tree -L 3
.
├── _book
│   ├── gitbook
│   │   ├── fonts
│   │   ├── gitbook.js
│   │   ├── gitbook-plugin-fontsettings
│   │   ├── gitbook-plugin-highlight
│   │   ├── gitbook-plugin-lunr
│   │   ├── gitbook-plugin-search
│   │   ├── gitbook-plugin-sharing
│   │   ├── images
│   │   ├── style.css
│   │   └── theme.js
│   ├── index.html
│   └── search_index.json
├── README.md
└── SUMMARY.md
```

5. 安装插件

1. 插入logo

在书籍的配置文件（后续有配置相关文章）配置好需要的 Gitbook 插件，执行 `gitbook install` 就可以在线安装相关插件了。

```
$ gitbook install
info: installing 15 plugins using npm@3.9.2
info:
info: installing plugin "highlight"
......
```

出现这样的输出后，就等待插件安装完成就可以使用了，如果插件配置错误，会中途直接报错停止，需要解决错误后才可以继续安装。

6. 本地静态电子书预览

使用 `gitboo serve` 命令开启进行本地网页预览服务，执行后会默认执行 `gitbook build` 命令，然后本地开启一个端口 `4000` 的预览网页服务，此时可以通过浏览器访问本地机器 4000 端口进行电子书的浏览。

1. 插入logo

```
$ gitbook serve
Live reload server started on port: 35729
Press CTRL+C to quit ...

info: 21 plugins are installed
info: 19 explicitly listed
info: loading plugin "highlight"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "code"... OK
info: loading plugin "splitter"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "github"... OK
info: loading plugin "alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "popup"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "donate"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "livereload"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 3 pages
info: found 0 asset files
warn: "options" property is deprecated, use config.get(key) instead
info: >> generation finished with success in 0.7s !

Starting server ...
Serving book on http://localhost:4000
```

此时通过浏览器打开 `http://localhost:4000` 即可预览静态网页电子书。

预览效果：

Image 2.2.1 - 预览效果图

好了，这就是 `Gitbook` 的主要的几个主要命令，如果需要了解更多，请参考相关网站。

1. 简单介绍

本章来介绍一下 `Gitbook` 的目录结构，下面的目录结构表示了一个简单的 Gitbook 的目录结构。

```
.
├── book.json
├── README.md
├── SUMMARY.md
├── GLOSSARY.md
├── chapter-1/
|   ├── README.md
|   └── something.md
└── chapter-2/
    ├── README.md
    └── something.md
```

GitBook 目录文件的主要功能：

文件名	描述
book.json	配置数据 (可选)
README.md	电子书的前言或简介 (必需)
SUMMARY.md	电子书目录 (可选)
GLOSSARY.md	词汇/注释术语列表 (可选)

2. 菜单结构

1. 插入logo

`SUMMARY.md` 文件描述了书籍的菜单结构。

1. `[]` 指定菜单项目的标题
2. `()` 指定菜单文章文件的路径
3. 支持子目录的方式，章节和子章节用 `两个` 、 `四个空格` 或者 `tab` 键来分级
4. `#` 或者 `---` 进行不同 `Part` 的分类，分别由标题或者水平分割线方式表示不同的部分
5. 区域导航定位，在章节 `路径 md` 文件结尾使用 `#` 号加上文章内容中章节的标题就能实现区域导航

```
# Summary

### Part I
* [Part I](part1/README.md)
    * [Writing is nice](part1/README.md#writing)
    * [GitBook is nice](part1/README.md#gitbook)

### Part II
* [Part II](part2/README.md)
    * [We love feedback](part2/README.md#feedback)
    * [Better tools for authors](part2/README.md#tools)

----
* [Last part without title](part3/title.md)
```

3. 页面文件

`Gitbook` 书籍的页面文件采用 `Markdown` 的语法实现，电子书的第一页内容是从文件 `README.md` 中提取的。如果这个文件名没有出现在 `SUMMARY` 中，那么它会被添加为章节的第一个条目。对 Markdown 语法不熟悉的可以参考：菜鸟教程、MarkDown中文网 或者其他参考的网站。

参考示例：

```
# Title of the chapter

This is a great introduction.

## Section 1

Markdown will dictates _most_ of your **book' s structure**

## Section 2

...
```

页面顶部描述

它使用 YAML 格式的风格来定义文档的描述信息，在 `三条虚线之间` ，文档中也可以不写顶部描述，这个不是必须的。

参考示例：

```
---
description: This is a short description of my page
---

# The content of my page
...
```

4. 专业术语列表

在 Gitbook 中使用 `GLOSSARY.md` 来进行 `专业术语列表的配置` 。将一些专业名词，名词或者术语的解释配置定义在文件中，在书籍中使用到对应专业术语的地方就可以链接到专业解释的地方。

定义的方式是在 GLOSSARY.md 使用 `##` 列表来定义专业术语的列表。

参考示例：

```
## markdown
Markdown是一种轻量级标记语言，创始人为约翰·格鲁伯（英语：John Gruber）。
它允许人们使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML（或者HTML）文档。
这种语言吸收了很多在电子邮件中已有的纯文本标记的特性。

## gitbook
GitBook 是一个基于 Node.js 的命令行工具，可使用 Github/Git 和 Markdown 来制作精美的电子书。

## Term
Definition for this term

## Another term
With it' s definition, this can contain bold text
and all other kinds of inline markup ...
```

5. 忽略文件

在 `SUMMARY.md` 中未列出的文件。所有静态文件，包含图片、JS、CSS都会复制到对应目录下，对于一些不需要的文件，GitBook将读取 `.gitignore` 、 `.bookignore` 和 `.ignore` 文件，以获取要忽略的文件和文件夹的列表。被忽略的文件不会被上传到版本中。这些文件的语法和 `Git` 中的 gitignore 语法相同。

参考示例：

```
# This is a comment

# Ignore the file test.md
test.md

# Ignore everything in the directory "bin"
bin/*
```

6. 以子目录的方式与项目集成

对于 Gitbook 书籍项目，可以使用 `子目录` (如example-docs/)来存储项目的文档。您可以在 `book.json` 中通过配置选项告诉 `GitBook` 在那里找到根目录

> 注意： `book.json` 文件除外， book.json 文件所在位置代表的就是项目的根目录，但是可以将书籍项目的其他文件放置在子目录。

1. 插入logo

参考示例：

```
1. 项目目录结构：
.
├── book.json
└── example-docs/
    ├── README.md
    └── SUMMARY.md

2. book.json 中的配置：
{
    "root": "./example-docs"
}
```

7. 总结

本章介绍了 `Gitbook` 中的基本目录的结构和对应的用处。熟悉后可以对 Gitbook 的体系结构更加熟悉，同时也能更好的去书写你的书籍了。

参考：官网介绍

1. 简述

本章主要介绍一下 `Gitbook` 中的相关配置以及说明。

Gitbook 使用了可选的 `JSON` 格式的配置文件来自定义书籍和文档的配置，这些配置选项通过 Gitbook项目根目录下的 `book.json` 文件来进行配置和指定。如果对 **JSON** 语法不熟悉的读者，可以参考：https://www.w3school.com.cn/json/json_syntax.asp 。

2. 配置概览

变量	说明
`root`	包含所有图书文件的根文件夹的路径，除了 `book.json`
`title`	书籍的标题，默认值从README中提取
`description`	您的书籍说明，默认值从自述文件中提取
`author`	作者姓名
`isbn`	书籍的国际码ISBN
`language`	ISO 语言规范中的语言规范定义，默认值是 `en`
`direction`	文本的方向，可以是 `rtl` 或 `ltr` ，默认值取决于 `language` 的值
`gitbook`	GitBook的版本，使用SemVer规范并接受诸如 `">=3.0.0"` 的条件
`structure`	指定自述，摘要，词汇表等的路径
`variables`	这个选项定义书籍中的变量
`links`	在左侧导航栏添加指定的链接信息
`styles`	这个选项是用来自定义书本的css的
`plugins`	指定书籍使用的插件列表
`pluginsConfig`	配置指定插件的一些配置信息

3. 配置示例

3.1. 基础配置

`Gitbook` 基础的配置信息，通过下面的这个方式直接配置在 `book.json` 中。

参考示例：

1. 插入logo

```
{
    "root":".",
    "author":"JiangMing",
    "title":"JiangMing Gitbook",
    "language":"zh-hans",
    "description":"This is gitbook",
    "isbn":"000-0-00-000000-0",
    "direction":"ltr",
    "gitbook":">=3.2.3"
}
```

3.2. structure 结构配置

除了 `root` 变量，你可以通过 `structure` 告诉Gitbook [Readme]，[Summary]，[Glossary]，[Languages]的文件名(而不是使用默认名称，如README.md)。这些文件必须在您的书籍项目的根目录。不接受像 `doc/README.md` 这样的子目录路径。

变量	说明
`structure.readme`	自述文件名(默认为“README.md”)
`structure.summary`	摘要文件名(默认为“SUMMARY.md”)
`structure.glossary`	词汇表文件名(默认为“GLOSSARY.md”)
`structure.languages`	语言文件名(默认为LANGS.md)

参考示例：

```
{
    "structure":{
        "readme":"README.md",
        "summary":"SUMMARY.md",
        "glossary":"GLOSSARY.md",
        "languages":"LANGS.md" // 注意默认已经配置，在 gitbook-V3.2.3版本配置会报错，可
    }
}
```

3.3. variables 变量配置

1. 插入logo

定义一些书籍中的变量信息，定义在 `book.json` 中的变量可以在 `book` 作用域下被访问，如：`{{ book.blog }}` 双括号语法在 **书籍中** 获取其中的数值。

参考示例：

```
{
    "variables":{
        "blog":"https://blog.csdn.net/ming_97y"
    }
}
```

3.4. links 链接导航

通过 `links` 配置在左侧导航栏添加指定的链接导航，如：添加自己的**博客链接**，**GIthub链接**等等...

参考示例：

```
{
    "links":{
        "sidebar":{
            "Blog":"https://blog.csdn.net/ming_97y",
            "Github":"https://github.com/jiangminggithub"
        }
    }
}
```

3.5. styles 自定义样式

通过 `styles` 配置这个选项用来自定义书本的 `css` 的。

参考示例：

1. 插入logo

```
{
    "styles": {
        "website": "styles/website.css",
        "ebook": "styles/ebook.css",
        "pdf": "styles/pdf.css",
        "mobi": "styles/mobi.css",
        "epub": "styles/epub.css"
    }
}
```

3.6. pdf 参数配置

PDF输出可以使用book.json中的 `pdf` 来进行配置：

变量	说明
`pdf.pageNumbers`	将页码添加到每个页面的底部(默认为true)
`pdf.fontSize`	基本字体大小(默认为12)
`pdf.fontFamily`	基本字体系列(默认为Arial)
`pdf.paperSize`	纸张大小，选项为： `"a0"，"a1"，"a2"，"a3"，"a4"，"a5"，"a6"，"b0"，"b1"，"b2" "b4"，"b5"，"b6"，"legal"，"letter"，(默认为"a4")`
`pdf.margin.top`	顶部边距(默认为56)
`pdf.margin.bottom`	底边距(默认为56)
`pdf.margin.right`	右边距(默认为62)
`pdf.margin.left`	左边距(默认为62)

参考示例：

1. 插入logo

```
{
    "pdf":{
        "pageNumbers":true,
        "fontFamily":"Arial",
        "fontSize":12,
        "paperSize":"a4",
        "margin":{
            "right":62,
            "left":62,
            "top":56,
            "bottom":56
        }
    }
}
```

3.7. plugins 插件列表

通过 `plugins` 配置可以配置书籍需要的插件列表。

参考示例：

```
{
    "plugins": [
        "github",
        "splitter",
        ...
    ]
}
```

3.8. pluginsConfig 插件配置

通过 `插件配置` 可以配置插件列表 `plugins` 中对应插件的一些配置选项信息。

参考示例：

1. 插入logo

```
{
    "plugins": ["github"],
    "pluginsConfig": {
        "github": {
            "url": "https://github.com"
        }
    }
}
```

4. 参考配置示例

下面的这个是一个简单的 `book.json` 的配置，可供参考。

1. 插入logo

```
{
    "root": ".",
    "author": "JiangMing",
    "title": "JiangMing Gitbook",
    "language": "zh-hans",
    "description": "This is gitbook",
    "isbn": "000-0-00-000000-0",
    "direction": "ltr",
    "gitbook": ">=3.2.3",
    "structure": {
        "readme": "README.md",
        "summary": "SUMMARY.md",
        "glossary": "GLOSSARY.md"
    },
    "variables": {
        "blog": "https://blog.csdn.net/ming_97y"
    },
    "links": {
        "sidebar": {
            "Blog": "https://blog.csdn.net/ming_97y",
            "Github": "https://github.com/jiangminggithub"
        }
    },
    "styles": {
        "website": "styles/website.css",
        "ebook": "styles/ebook.css",
        "pdf": "styles/pdf.css",
        "mobi": "styles/mobi.css",
        "epub": "styles/epub.css"
    },
    "pdf": {
        "pageNumbers": true,
        "fontFamily": "Arial",
        "fontSize": 12,
        "paperSize": "a4",
        "margin": {
            "right": 62,
            "left": 62,
            "top": 56,
            "bottom": 56
        }
    },
    "plugins": [
        "-lunr",
        "-search",
        "advanced-emoji",
        "search-plus",
        "github",
        "splitter",
        "anchor-navigation-ex",
```

```
            "chapter-fold",
            "expandable-chapters-small",
            "code",
            "alerts",
            "insert-logo",
            "flexible-alerts"
        ],
        "pluginsConfig": {
            "github": {
                "url": "https://github.com"
            },
            "insert-logo": {
                "url": "jim-logo.png",
                "style": "background: none; max-height: 100px; min-height: 30px"
            },
            "flexible-alerts": {
                "style": "callout",
                "comment": {
                    "label": "Comment",
                    "icon": "fa fa-comments",
                    "className": "info"
                }
            }
        }
    }
```

5. 总结

到此就完成了 `Gitbook` 中关于配置文件的相关介绍，了解玩这些配置，就可以更好的去配置和使用 Gitbook 来完成自己的电子书籍的书写了。

参考： 官网说明

1. 介绍

本章主要来详细的介绍一下 `Gitbook` 中的 **插件** 相关的配置和使用。在 Gitbook 中可以在书籍的配置文件 `book.json` 中来进行插件的相关配置。比如有很多好用的插件，可以很好的拓展书籍的外观，可用性或者其他方便的使用，所以接下来就一起来看看 Gitbook 中插件的使用吧。

配置的方法是在配置文件的 `plugins` 中添加需要的插件名称即可。

如下所示：

```
"plugins": [
    "search",
    "highlight",
    "sharing",
    "font-settings",
    "livereload",
    ...
]
```

2. 默认插件

Gitbook 中默认带有 5 个插件：

名称	说明
`highlight`	语法高亮插件，代码高亮功能
`search`	搜索插件，不支持中文搜索
`sharing`	分享插件，右上角分享功能
`font-settings`	字体设置（最上方的"A"符号）
`livereload`	热加载插件，为 GitBook 编辑进行实时重新预览加载

3. 禁用自带的插件

如果需要 `去除` 或者 `禁用` Gitbook 中的某个插件，可以在插件名称前面加 `-` 。

如下所示：

```
"plugins": [
    "-search",
    "-highlight",
    "-sharing",
    "-font-settings",
    "-livereload",
    ...
]
```

4. 添加插件列表

如果需要添加一些**第三方的自定义插件**，可以在 `plugins` 中添加需要的插件名称列表。

> [!WARNING]
>
> - 有的第三方的插件可能和默认的插件有重复，或者替代默认插件的，需要禁用对应的默认插件，具体用法一般参考对应插件的使用说明。
> - 第三方插件使用的话，可能会破坏书籍的结构，所以使用上需要注意！

例如：

1. 插入logo

```
"plugins": [
        "-search",
        "advanced-emoji",
        "search-pro",
        "github",
        "splitter",
        "anchor-navigation-ex",
        "chapter-fold",
        "expandable-chapters-small",
        "code",
        "alerts",
        "insert-logo",
        "flexible-alerts",
        ...
    ]
```

5. 插件属性配置 pluginsConfig

配置 `插件的属性` 在书籍配置文件中的 pluginsConfig 中进行相关插件的属性配置。

例如：配置 `insert-logo` 插件的相关属性

```
"pluginsConfig": {
        "insert-logo": {
            "url": "jim-logo.png",
            "style": "background: none; max-height: 100px; min-height: 30px"
        }
    }
```

6. 总结

到此将插件的配置和使用详细的介绍完了，也列举了一些常用的一些第三方的插件的使用方法，相信现在大家可以很好的利用这些比较好用的第三方的插件，去更好的去构建自己的书籍了。

实用插件

`Gitbook` 中的 `第三方插件` 有很多，在此就不进行一一的介绍了。下面就根据博主了解或者使用过的插件，简单整理一些实用的插件进行介绍一下吧。

第三方插件使用方法：

- 在配置文件 `book.json` 中添加 `"plugins"` 和 `"pluginConfig"` 字段，然后执行 `gitbook install` 来进行插件的安装
- 使用NPM安装 `npm install gitbook-plugin-插件名` 进行安装
- 从源码 `GitHub` 地址中下载，放到 `node_modules` 文件夹里安装

> [!TIP|style:flat]
>
> 1. 推荐使用配置文件 `book.json` 配置的方法进行安装，下面主要通过这种方式来进行介绍安装和配置第三方的插件的简单使用。
> 2. **更详细的配置或者使用方法，以及效果图，请参考每一个插件后面贴出官方参考链接**。

insert-logo 插入logo

将自己的 `logo` 图片插入到导航栏上方中，可以定制显示自己的 logo 标识。

配置使用方法：

```
{
    "plugins": [ "insert-logo" ],
    "pluginsConfig": {
      "insert-logo": {
        "url": "./jim-logo.png",
        "style": "background: none; max-height: 100px; min-height: 30px"
      }
    }
}
```

插件 Github 地址：https://github.com/matusnovak/gitbook-plugin-insert-logo

效果预览： 参考本书左上角logo图标

favicon 更改网站的图标

`自定义的网站图标` 的插件，可以将自己的 logo 图标设置为网站的图标。

配置使用方法：

```
{
    "plugins": [
        "favicon"
    ],
    "pluginsConfig": {
        "favicon": {
            "shortcut": "assets/images/favicon.ico",
            "bookmark": "assets/images/favicon.ico",
            "appleTouch": "assets/images/apple-touch-icon.png",
            "appleTouchMore": {
                "120x120": "assets/images/apple-touch-icon-120x120.png",
                "180x180": "assets/images/apple-touch-icon-180x180.png"
            }
        }
    }
}
```

插件 Github 地址：https://github.com/menduo/gitbook-plugin-favicon

效果预览：参考本书的网页标签图标

search-pro 高级搜索（支持中文）

支持中文搜索的插件, 使用此插件需要将默认的 `search` 和 `lunr` 插件去掉。

配置使用方法：

```
{
    "plugins": [
      "-lunr", "-search", "search-pro"
    ]
}
```

插件 Github 地址：https://github.com/gitbook-plugins/gitbook-plugin-search-pro

效果预览：

Image 4.2.3.1 - 高级搜索效果预览图

splitter 侧边栏宽度可调节

`splitter` 插件可以使左侧的 `侧边栏目录宽度` 可以自定义的调节。

配置使用方法：

```
{
    "plugins": ["splitter"]
}
```

插件 Github 地址：https://github.com/yoshidax/gitbook-plugin-splitter

效果预览： 参考本书的左侧边栏，可以调节宽度

github 在右上角添加github图标

`github` 插件会在右上角添加一个 github 的图标，可以通过插件属性配置链接，点击后可以进入自定义的链接页面。

配置使用方法：

```
{
    "plugins": [
        "github"
    ],
    "pluginsConfig": {
        "github": {
            "url": "https://github.com/jiangminggithub"
        }
    }
}
```

插件 Github 地址：https://github.com/GitbookIO/plugin-github

效果预览：参考本书的右上角 `github` 图标

sharing-plus 分享当前页面

分享当前页面的插件，比默认的 `sharing` 插件多了一些分享方式，同样可以通过配置插件属性进行相关的配置，可以通过实际需要进行相关配置。

配置使用方法：

```
{
    "plugins": ["-sharing", "sharing-plus"],
    "pluginsConfig": {
        "sharing": {
            "douban": false,
            "facebook": true,
            "google": false,
            "hatenaBookmark": false,
            "instapaper": false,
            "line": false,
            "linkedin": true,
            "messenger": false,
            "pocket": true,
            "qq": false,
            "qzone": false,
            "stumbleupon": false,
            "twitter": true,
            "viber": false,
            "vk": false,
            "weibo": false,
            "whatsapp": false,
            "all": [
                "facebook", "google", "twitter",
                "weibo", "instapaper", "linkedin",
                "pocket", "stumbleupon"
            ]
        }
    }
}
```

插件参考地址：https://www.npmjs.com/package/gitbook-plugin-sharing-plus

效果预览：参考本书的右上角 `分享` 图标

copy-code-button 代码复制按钮

为代码块添加一个可以 `复制的按钮` 。

配置使用方法：

```
{
    "plugins": ["copy-code-button"]
}
```

插件 Github 地址：https://github.com/WebEngage/gitbook-plugin-copy-code-button

效果预览：

Image 4.2.7.1 - 代码块拷贝按钮效果预览图

code 代码添加行号&复制按钮

这个插件可以为代码块添加 `行号` 和 `复制按钮` ，单行代码情况无行号。

如果需要 `去除代码复制按钮` ，可在配置文件进行配置 `copyButtons` 属性为 false。

配置使用方法：

```
{
    "plugins" : [
            "code"
     ],
    "pluginsConfig": {
      "code": {
        "copyButtons": false
      }
    }
}
```

插件 Github 地址：https://github.com/TGhoul/gitbook-plugin-code

效果预览：

```
{
    1,
    2,
    3,
    ...
}
```

advanced-emoji 表情图标

这个插件可以在书籍中使用表情列表中的表情图标。

配置使用方法：

```
{
    "plugins": [
        "advanced-emoji"
    ]
}
```

插件 Github 地址：https://github.com/codeclou/gitbook-plugin-advanced-emoji

效果预览：

emphasize 文字底色

这个插件可以给书籍中的 `文本内容` 加上特定的 `文字底色效果` 。

配置使用方法：

```
{
    "plugins": ["emphasize"]
}
```

简单的使用示例（markdown书籍中内容中）：

```
This text is {% em %}highlighted !{% endem %}

This text is {% em %}highlighted with **markdown**!{% endem %}

This text is {% em type="green" %}highlighted in green!{% endem %}

This text is {% em type="red" %}highlighted in red!{% endem %}

This text is {% em color="#ff0000" %}highlighted with a custom color!{% endem %}
```

插件 Github 地址：https://github.com/GitbookIO/plugin-emphasize

效果预览：

This text is highlighted !

This text is highlighted with **markdown**!

This text is highlighted in green!

This text is highlighted in red!

This text is highlighted with a custom color!

image-captions 在图片下面显示标题

抓取内容中图片的 `alt` 或 `title` 属性，在图片下面显示标题。更多详细的配置属性和使用方法参考官方说明。

配置使用方法：

```
{
    "plugins": [
        "image-captions"
    ],
    "pluginsConfig": {
      "image-captions": {
          "caption": "Image _PAGE_LEVEL_._PAGE_IMAGE_NUMBER_ - _CAPTION_",
          "align": "left",
          ...
      }
  }
}
```

插件 Github 地址：https://github.com/todvora/gitbook-plugin-image-captions

效果预览：

Image 4.2.11.1 - 显示图片信息预览图

anchor-navigation-ex 悬浮目录和回到顶部

插件功能：

- 给页面H1-H6标题增加锚点效果
- 浮动导航模式
- 页面内顶部导航模式
- 导航标题前的层级图标是否显示，自定义H1-H3的层级图标
- plugins["theme-default"],页面标题层级与官方默认主题的showLevel层级关联
- plugins["theme-default"],插件样式支持官网默认主题的三种样式：White、Sepia、Night
- 在页面中增加 `<extoc></extoc>` 标签，会在此处生成TOC目录
- 在页面中增加 `<!-- ex_nonav -->` 标签，不会在该页面生成悬浮导航
- config.printLog=true,打印当前的处理进度，排错很有用
- config.multipleH1=false,去掉丑陋的多余的1. 序号（如过您的书籍遵循一个MD文件只有一个H1标签的话）
- config.showGoTop=true,显示返回顶部按钮 V1.0.11+
- config.float.floatIcon 可以配置浮动导航的悬浮图标样式 V1.0.12+
- 在页面中增加 `<!-- ex_nolevel -->` 不会在该页面生成层级序号 V1.0.12+

配置使用方法：

```
{
  "plugins": [
       "anchor-navigation-ex"
  ]
}
```

插件 Github 地址：https://github.com/zq99299/gitbook-plugin-anchor-navigation-ex

效果预览：

alerts 漂亮格式的提示块

这个插件可以将将 `块引用` 转换为 `漂亮的提示格式` 的信息。

配置使用方法：

```
{
    "plugins": ["alerts"]
}
```

目前支持 4 种提示的类型： `info` ， `warning` ， `danger` ， `success`

```
Info styling

> **[info] For info**
>
> Use this for infomation messages.

Warning styling

> **[warning] For warning**
>
> Use this for warning messages.

Danger styling

> **[danger] For danger**
>
> Use this for danger messages.

Success styling

> **[success] For info**
>
> Use this for success messages.
```

插件参考地址：https://www.npmjs.com/package/gitbook-plugin-alerts

效果预览：

Info styling：

1. 插入logo

> **[info] For info**
>
> Use this for infomation messages.

Warning styling：

> **[warning] For warning**
>
> Use this for warning messages.

Danger styling：

> **[danger] For danger**
>
> Use this for danger messages.

Success styling：

> **[success] For info**
>
> Use this for success messages.

flexible-alerts 高级格式显示的提示块

这个插件将块引用转换为漂亮的警报。可以在全局和警报特定级别配置外观，因此输出确实符合您的需求。此外，您还可以提供自己的警报类型（比如最后的comment）。

配置使用方法：

```
{
    "plugins": [
      "flexible-alerts"
    ],
    "pluginsConfig": {
      "flexible-alerts": {
        "style": "callout",
        "comment": {
          "label": "Comment",
          "icon": "fa fa-comments",
          "className": "info"
        }
      }
    }
}
```

用法：

```
> [!type| style:xx| label:xx| icon:xx| className:xx| labelVisibility:xx| iconVisibility
> 内容部分
```

字段介绍，如果不设置的表示选择默认，除了!type都不是必需的。

1. 插入logo

键	允许的值	说明
!type	NOTE，TIP，WARNING和DANGER	警告级别设置
style	以下值之一: callout（默认），flat	警告样式，见图19的左右不同
label	任何文字	警告块的标题位置，即Note这个字段位置（不支持中文）
icon	e.g. 'fa fa-info-circle'	一个有效的Font Awesome图标，那块小符号
className	CSS类的名称	指定css文件，用于指定外观
labelVisibility	以下值之一：visible（默认），hidden	标签是否可见
iconVisibility	以下值之一：visible（默认），hidden	图标是否可见

```
1. 这是简单的用法
> [!NOTE]
> 这是一个简单的Note类型的使用，所有的属性都是默认值。

---
2. 这是自定义属性的用法
> [!NOTE| style:flat| lable:Mylable| iconVisibility:hidden]
> "!type":`NOTE`、"style":`flat`、"lable":`自定义标签`、图标不可见
```

`json` 配置个性化，自定义一个COMMENT类型使用。

```
"pluginsConfig": {
      "flexible-alerts": {
        "style": "callout",
        "comment": {
          "label": "Comment",
          "icon": "fa fa-comments",
          "className": "info"
        }
      }
    }
```

1. 插入logo

使用：

```
> [!COMMENT]
> An alert of type 'comment' using style 'callout' with default settings.
```

插件 Github 地址：https://github.com/fzankl/gitbook-plugin-flexible-alerts

效果预览：

1. 简单的使用效果：

Note:

> [!NOTE] 这是一个简单的Note类型的使用，所有的属性都是默认值。

TIP:

> [!TIP] 这是一个简单的Note类型的使用，所有的属性都是默认值。

WARNING:

> [!WARNING] 这是一个简单的Note类型的使用，所有的属性都是默认值。

DANGER:

> [!DANGER] 这是一个简单的Note类型的使用，所有的属性都是默认值。

1. 自定义属性效果：

> [!NOTE|style:flat|lable:Mylable|iconVisibility:hidden]
> "!type": `NOTE` 、"style": `flat` 、"lable": `自定义标签` 、图标不可见

1. 个性化使用效果：

> [!COMMENT] An alert of type 'comment' using style 'callout' with default settings.

auto-scroll-table 自动表格过长滚动条

为避免表格过宽，在 `过宽的表格页面中增加滚动条` 。

配置使用方法：

```
{
  "plugins": ["auto-scroll-table"]
}
```

效果预览：

标题	
11111111111111111111111111111111111111111111111111111111	222222
44444444444444444444444444444444444444444444444444444444	555555

lightbox 点击图片弹窗显示

这个插件可以单击图片，以图片 `本身大小的方式弹窗显示` 图片和一些图片相关的 `Alt` 的信息。

配置使用方法：

```
{
    "plugins": [
        "lightbox"
    ],
    "lightbox": {
        "includeJQuery": false,
        "sameUuid": true,
        "options": {
            "resizeDuration": 500,
            "wrapAround": false
        }
    }
}
```

配置参数介绍：

- `includeJQuery`：如果你的项目中已经引入了 `JQuery` 可以在此设置是否包含插件本身的 JQuery。
- `sameUuid`：在图片预览中添加上一个、下一个按钮来浏览本页面的图片配置。
- `options`： 这个选项配置显示的动画时长，是否包裹等相关配置。

插件 Github 地址：https://github.com/vongola12324/gitbook-plugin-lightbox

效果预览：

Image 4.2.16.1 - 点击查看图片

popup 点击图片新页面弹出显示

可以单击图片，在 `新页面查看大图显示` 。

配置使用方法：

```
{
  "plugins": [ "popup" ]
}
```

插件 Github 地址：https://github.com/somax/gitbook-plugin-popup

效果预览：

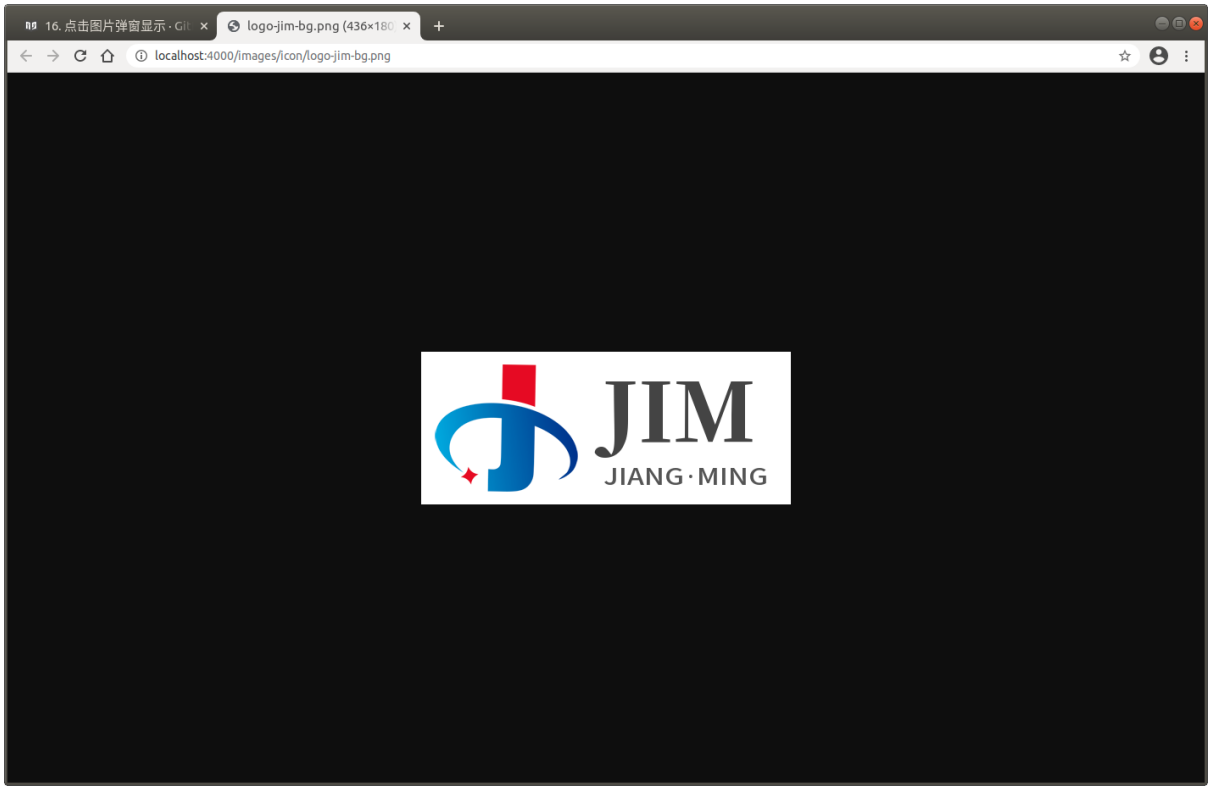

Image 4.2.17.1 - 点击图片新页面打开预览图

tbfed-pagefooter 添加页脚和版权

可以添加 `页脚，版权信息` 。

配置使用方法：

```
{
    "plugins": [
       "tbfed-pagefooter"
    ],
    "pluginsConfig": {
        "tbfed-pagefooter": {
            "copyright":"Copyright ©JiangMing",
            "modify_label": "更新时间: ",
            "modify_format": "YYYY-MM-DD HH:mm:ss"
        }
    }
}
```

如果想自定义一个链接说明，自己可以去index.js里，把 `powered by gitbook` ，改成 `powered by <a href="你的说明内容链接" target="_blank">` 你的说明内容

插件 Github 地址：https://github.com/zhj3618/gitbook-plugin-tbfed-pagefooter

效果预览，参考本页尾的【页脚和版权】效果。

Copyright ©JiangMing all right reserved，powered by JiangMing-JIM更新时间：2021-03-28 21:20:22

page-footer 高级页脚和版权

`更高级的页脚版权信息的插件`，支持 `normal`， `symmetrical`， `Issues` 三种不同样式的形式。

配置使用方法：

```
{
    "plugins": [
        "page-footer"
    ],
    "pluginsConfig": {
        "page-footer": {
            "description": "modified at",
            "signature": "Aleen",
            "wisdom": "More than a coder, more than a designer",
            "format": "yyyy-MM-dd hh:mm:ss",
            "copyright": "Copyright &#169; aleen42",
            "timeColor": "#666",
            "copyrightColor": "#666",
            "utcOffset": "8",
            "isShowQRCode": true,
            "baseUri": "https://aleen42.gitbooks.io/personalwiki/content/",
            "isShowIssues": true,
            "repo": "aleen42/PersonalWiki",
            "issueNum": "8",
            "token": "",
            "style": "normal"
        }
    }
}
```

插件 Github 地址：https://github.com/aleen42/gitbook-footer

效果预览：

Normal效果：

1. 插入logo

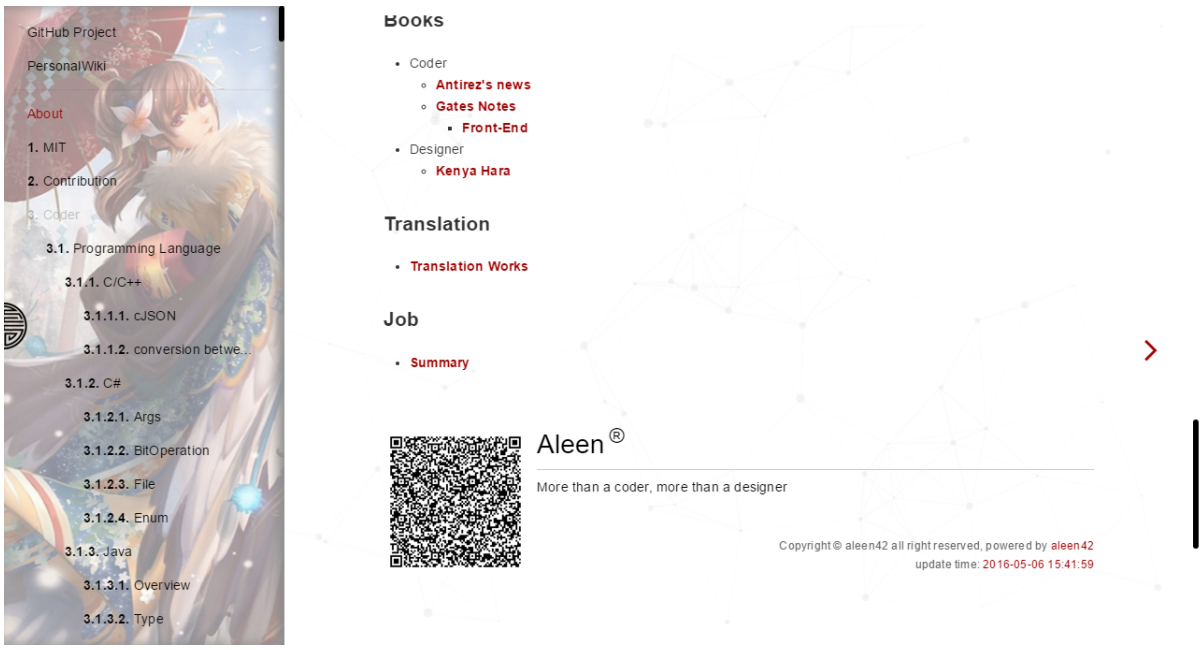

Image 4.2.19.1 - Normal效果预览

Symmetrical效果：

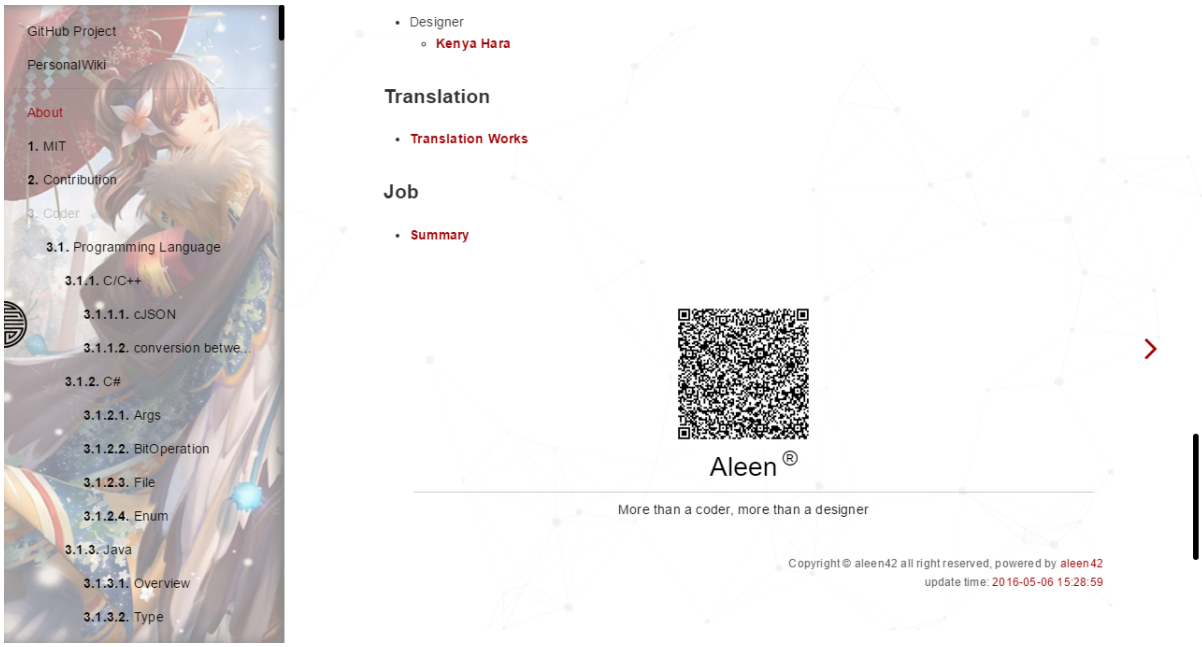

Image 4.2.19.2 - Symmetrical效果预览

Issues效果：

1. 插入logo

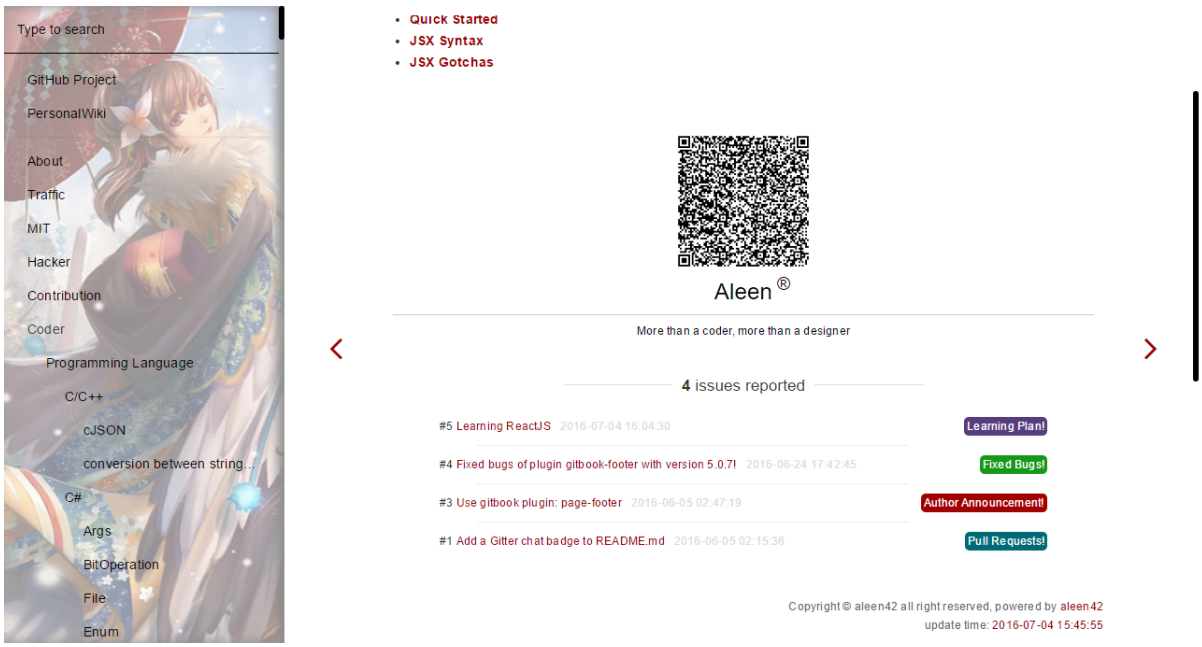

Image 4.2.19.3 - Issues效果预览

hide-element 隐藏元素

可以隐藏不想看到的元素，比如导航栏中 `Published by GitBook` 。

配置使用方法：

```
{
    "plugins": [
        "hide-element"
    ],
    "pluginsConfig": {
        "hide-element": {
            "elements": [".gitbook-link"]
        }
    }
}
```

插件 Github 地址：https://github.com/gonjay/gitbook-plugin-hide-element

效果预览：

Image 4.2.20.1 - 隐藏元素预览

back-to-top-button 回到顶部按钮

可以在浏览文章到 `一定长度` 的时候，显示一个 `回到顶部的快捷按钮` ，点击可以快速回到文章顶部。

配置使用方法：

```
{
    "plugins": [
        "back-to-top-button"
    ]
}
```

插件 Github 地址：https://github.com/stuebersystems/gitbook-plugin-back-to-top-button

效果预览：

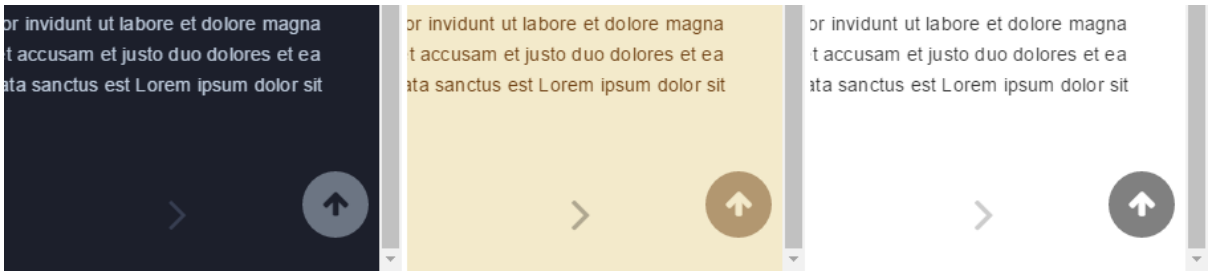

Image 4.2.21.1 - 回到顶部按钮预览

prism 基于 Prism 的代码高亮

`prism` 基于 Prism 的代码高亮插件，可以为代码块配置不同的主题风格。

配置使用方法：

```
{
    "plugins": [
        "prism",
        "-highlight"
    ],
    "pluginsConfig": {
        "prism": {
            "css": [
                "prismjs/themes/prism-solarizedlight.css"
            ],
            "lang": {
                "flow": "typescript"
            },
            "ignore": [
                "mermaid",
                "eval-js"
            ]
        }
    }
}
```

配置参数介绍：

- `css`：指定自定义主题的样式文件。
- `lang`： 配置自定义语言前缀名，来混淆配置。
- `ignore`：由于其他插件使用自定 `lang` 代码块的概念来表示其他功能，你可以忽略某些 langs。

插件 Github 地址：https://github.com/gaearon/gitbook-plugin-prism

效果预览：

prism-okaidia.css预览：

```
class Polygon {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  get area() {
    return this.height * this.width;
  }
}
```

Image 4.2.22.1 - prism-okaidia.css预览

prism-solarizedlight.css预览：

```
class Polygon {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  get area() {
    return this.height * this.width;
  }
}
```

Image 4.2.22.2 - prism-solarizedlight.css预览

prism-tomorrow.css预览：

```
class Polygon {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
  get area() {
    return this.height * this.width;
  }
}
```

Image 4.2.22.3 - prism-tomorrow.css预览

更多主题效果参考：https://github.com/gaearon/gitbook-plugin-prism

theme-hqbook 自定义 hqbook 主题

`Gitbook` 也支持 `通过插件的方式来自定义主题`，在 Github中可能有很多主题资源，但是不是非常建议大家去使用第三方的主题，因为第三的主题可能会导致一些不兼容的情况或者各种问题。

> [!DANGER|style:flat|lable:特别注意] `hqbook` 主题其实也是不是非常完善的，比如不能很好的适配移动端，在移动端访问，界面就可能出现各种不协调的地方，截止到目前为止，笔者还没有发现一个比较完善的主题，本节仅供大家参考，具体的使用还是需要自己去研究或者有能力的情况自行开发一套哈。

配置使用方法：

1. 插入logo

```
{
    "plugins": [
        "theme-hqbook"
    ],
    "variables": {
        "themeHqbook":{
            "nav":[
                {
                    "url": "https://blog.csdn.net/ming_97y",
                    "target": "_blank",
                    "name": "Blog"
                },
                // { ... }
            ]
        },
    },
    "pluginsConfig": {
        "theme-hqbook":{
            "favicon": "./favicon.ico",
            "logo":"./logo.png",
            "search-placeholder":"输入关键字搜索",
            "copyButtons": true,
            "copyLines": true,
            "dragSplitter": true,
            "hide-elements": [
                ".summary .gitbook-link"
            ],
            "flexible-linkcard": {
                "title": "flexible-linkcard",
                "hrefUrl": "https://github.com/HaoqiangChen/gitbook-plugin-flexibl
                "target": "_blank",
                "imgSrc": "./book/logo.png",
                "imgClass": "rect"
            }
        }
    }
}
```

参数简单介绍：

- `favicon`：自定义favicon地址，修改标题栏图标
- `logo`：自定义logo地址，修改logo
- `search-placeholder`：搜索框提示信息
- `copyButtons`：代码块添加复制按钮
- `copyLines`：当显示多行代码时，将添加行号

1. 插入logo

- `dragSplitter`：在左侧目录和右侧内容之间添加一个可以拖拽的栏，用来调整两边的宽度
- `hide-elements`：隐藏元素，比如导航栏中Published by GitBook
- `nav`：顶部导航栏，nav为数组，将需要的导航添加到变量themeHqbook中
- `flexible-linkcard`：
- `title`：定义flexible-linkcard的默认标题
- `hrefUrl`：定义flexible-linkcard的默认网址
- `target`：定义flexible-linkcard的网址默认打开方式，即`HTML<a>的target属性，属性值有_seft, _blank, _parent, _top`几种，最好还是设置_blank新窗口打开
- `imgSrc`：定义flexible-linkcard的默认显示图片
- `imgClass`：定义flexible-linkcard的默认图片样式

插件 Github 地址：https://github.com/HaoqiangChen/gitbook-plugin-theme-hqbook

效果预览：

Image 4.2.23.1 - hqhook效果预览图

expandable-chapters-small 折叠侧边栏菜单

在左侧目录前面显示一个折叠的标志，可以进行 `折叠侧边栏` 。

实现这个功能的目前常见的有3个插件：

- chapter-fold：指示小图标
- expandable-chapters： 指示大图标
- expandable-chapters-small：指示小图标

配置使用方法：

```
{
    plugins: ["chapter-fold"]
}

或者
{
    plugins: ["expandable-chapters"]
}

或者
{
    plugins: ["expandable-chapters-small"]
}
```

插件 Github 地址：https://github.com/chrisjake/gitbook-plugin-expandable-chapters-small

效果预览：参考本书的左边菜单折叠效果

donate 打赏插件

显示在文章最下面的 `按钮` ，点击可弹出 `收款相关的图片和信息` 。

配置使用方法：

```
{
    "plugins": ["donate"],
    "pluginsConfig": {
        "donate": {
          "wechat": "/images/wechat-qr.png",
          "alipay": "/images/alipay-qr.png",
          "title": "默认空",
          "button": "默认值：Donate",
          "alipayText": "默认值：支付宝捐赠",
          "wechatText": "默认值：微信捐赠"
        }
    }
}
```

插件 Github 地址：https://github.com/willin/gitbook-plugin-donate

效果预览：参考页面下方的打赏按钮插件

更多插件参考

其他插件参考：

- ad 在每个页面顶部和底部添加广告或任何自定义内容：https://github.com/zhaoda/gitbook-plugin-ad
- sectionx 分离各个段落，并提供一个展开收起的按钮：https://github.com/ymcatar/gitbook-plugin-sectionx
- Puml 使用 PlantUML 展示 uml 图: https://github.com/GitbookIO/plugin-puml
- Graph 使用 function-plot 绘制数学函数图：https://github.com/cjam/gitbook-plugin-graph
- todo 添加 todo（选中框） 功能：https://github.com/ly-tools/gitbook-plugin-todo
- include-csv 展示 csv 文件内容：https://github.com/TakuroFukamizu/gitbook-plugin-include-csv
- musicxml 支持 musicxml 格式的乐谱渲染：https://github.com/ymcatar/gitbook-plugin-musicxml
- url-embed 嵌入动态内容：https://github.com/basilvetas/gitbook-plugin-url-embed
- ...

关于第三方插件的介绍就到这里了，大家有兴趣的可以去查看一下：https://github.com/GitbookIO

更多的插件，请到 NPM Package 或者 GitHub 中查询使用。

导出电子书

目前为止，Gitbook 支持如下输出：

- 静态HTML（静态网站）
- PDF格式
- eBook格式
- Mobi 格式

> [!TIP|style:flat]
>
> 目前常见的电子书格式主要主要有三种(`ePub` , `Mobi` , `PDF`)，在 `Gitbook` 中导出这三种格式都依赖于系统本身提供的 `ebook-convert` 工具支持.

在本书的 `安装章节中` 已经介绍了怎么安装 `ebook-convert` 工具，再这里就不多介绍了，接下来将针对上面的几种常见的电子书的导出进行介绍。

基本命令

> [!Note|style:flat]
>
> 语法格式:
>
> - gitbook build：导出静态网站HTML格式文件
> - gitbook pdf：导出PDF格式文件
> - gitbook epub：导出ePub格式文件
> - gitbook mobi：导出Mobi格式文件

简单示例：

1. 插入logo

```
# 1. 生成 `html` 静态网站文件并输出 `debug` 级别日志
$ gitbook build --log=debug

# 2. 生成 `pdf` 文件并输出 `debug` 级别日志
$ gitbook pdf book.pdf --log=debug

# 3. 生成 `epub` 文件并输出 `debug` 级别日志
$ gitbook epub book.epub --log=debug

# 4. 生成 `mobi` 文件并输出 `debug` 级别日志
$ gitbook mobi book.mobi --log=debug
```

导出书籍为HTML格式的静态网站

`Gitbook` 默认静态编译就是 HTML 格式的静态网站，默认导出文件到书籍目录的下的 `_book` 中。

在书籍目录终端中执行 `gitbook build` 命令就可以将书籍导出到目录 `_bo0k` 目录中HTML的静态网页HTML文件。

参考示例：

1. 插入logo

```
# 静态编译导出HTML静态网站文件，如果需要查看生成日志过程，可以加上--log=debug
$ gitbook build
info: 28 plugins are installed
info: 22 explicitly listed
info: loading plugin "insert-logo"... OK
info: loading plugin "favicon"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "splitter"... OK
info: loading plugin "github"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "code"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "emphasize"... OK
info: loading plugin "image-captions"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "alerts"... OK
info: loading plugin "flexible-alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "lightbox"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "prism"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "donate"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 38 pages
info: found 26 asset files
warn: "options" property is deprecated, use config.get(key) instead
warn: "options.generator" property is deprecated, use "output.name" instead
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
info: >> generation finished with success in 2.8s !

# 简单查看生成的_book目录
$ tree -L 1
.
├── _book
├── book.json
├── node_modules
├── README.md
└── SUMMARY.md

# 简单查看_book目录中的内容
$ cd  _book
$ tree -L 1
.
├── gitbook
```

```
├── index.html
└── search_plus_index.json
```

效果预览

Image 5.1.1.1 - 静态HTML文件效果预览图

导出书籍为PDF格式文件

相信大家对 `PDF` 格式的文件应该不会陌生的吧，在实际生活的场景中，PDF 文件也是比较常见的，比如电子发票，公告文档，材料文档，说明文档等等...

接下来就来介绍一下 `Gitbook` 中将书籍怎么导出成一个PDF格式的文件。在书籍目录终端中执行 `git book pdf` 命令就可以将书籍导出到书籍目录下的一个名称为 `book.pdf` 的文件了。

使用示例：

1. 插入logo

```
# 将书籍导出到PDF文件，如果需要查看生成日志过程，可以加上--log=debug
$ gitbook pdf
info: 28 plugins are installed
info: 22 explicitly listed
info: loading plugin "insert-logo"... OK
info: loading plugin "favicon"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "splitter"... OK
info: loading plugin "github"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "code"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "emphasize"... OK
info: loading plugin "image-captions"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "alerts"... OK
info: loading plugin "flexible-alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "lightbox"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "prism"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "donate"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 38 pages
info: found 28 asset files
warn: "options" property is deprecated, use config.get(key) instead
warn: "options.generator" property is deprecated, use "output.name" instead
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
info: >> generation finished with success in 11.9s !
info: >> 1 file(s) generated

# 查看生成的book.pdf文件
$ tree -L 1
.
├── _book
├── book.json
├── book.pdf
├── images
├── node_modules
├── README.md
└── SUMMARY.md
```

效果预览

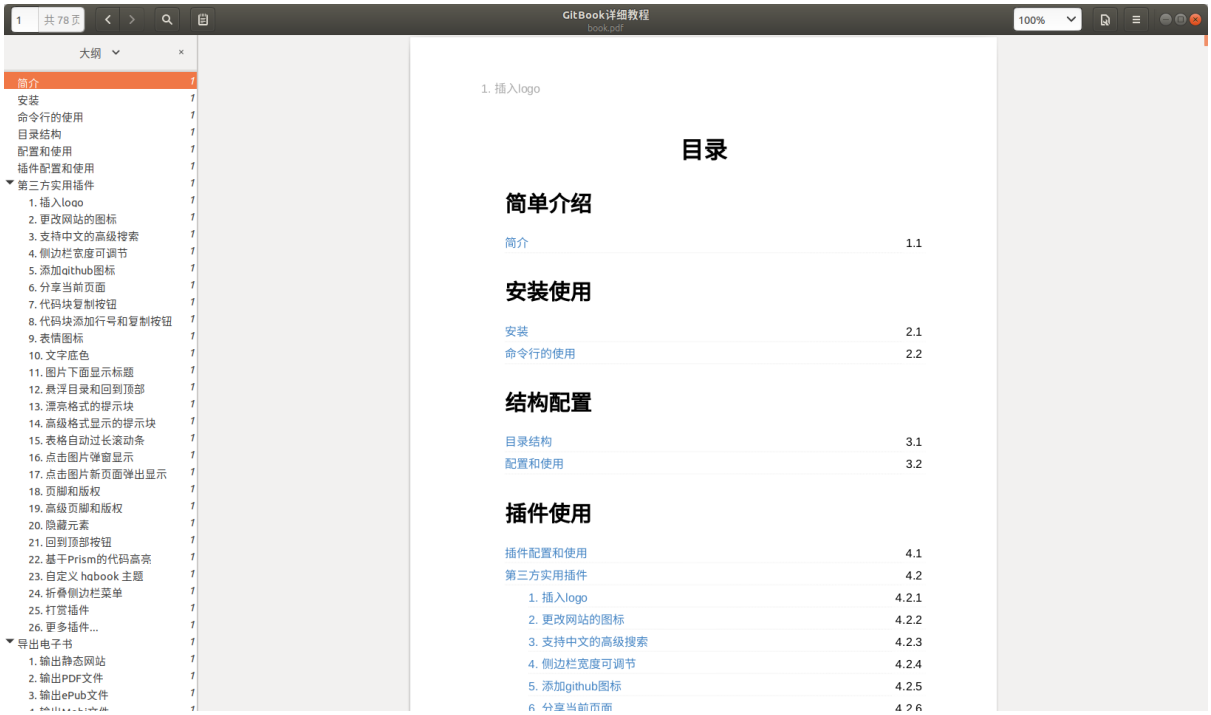

Image 5.1.2.1 - PDF格式文件效果图预览

导出书籍为ePub格式文件

说起 `ePub` 文件格式，相信大多数人还是比较不熟悉的，相比较 PDF 格式，ePub 格式在日常生活中使用的场景并没有那么平凡，所以这里简单给大家介绍一下 ePub 格式：

ePub（Electronic Publication的缩写，意为：电子出版），是一个自由的开放标准，属于一种可以“自动重新编排”的内容。

百度百科中的简介：

> [!Note|style:flat]
>
> 电子出版（Electronic Publishing）是指以数字代码方式将图、文、声、像等信息编辑加工后存储在磁、光、电介质上，信息通过计算机或其他具有类似功能的设备读取使用的一种出版形式。电子出版（elektronisches Publizieren）分成在线电子出版（elektronisches Online-Publizieren）和离线电子出版（elektronisches Offline-Publizieren）两大类型。
>
> 通常包括前期策划、素材准备、美术设计、程序编制、后期制作或通过网络发送等环节。可见电子出版以计算机为生产工具，原作的大量复制也是以计算机为核心。这里，定义出版时所必需的编辑改为前期策划、素材准备和美术设计等。因此，与传统意义上的出版相比，电子出版包含了更复杂的劳动。工艺手段和技术含量也更高。

在书籍目录终端中执行 `gitbook epub` 命令就可以生成一个名称为 `book.epub` 的文件。

使用示例：

1. 插入logo

```
# 将书籍导出到ePub文件，如果需要查看生成日志过程，可以加上--log=debug
$ gitbook epub
info: 28 plugins are installed
info: 22 explicitly listed
info: loading plugin "insert-logo"... OK
info: loading plugin "favicon"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "splitter"... OK
info: loading plugin "github"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "code"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "emphasize"... OK
info: loading plugin "image-captions"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "alerts"... OK
info: loading plugin "flexible-alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "lightbox"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "prism"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "donate"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 38 pages
info: found 29 asset files
warn: "options" property is deprecated, use config.get(key) instead
warn: "options.generator" property is deprecated, use "output.name" instead
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
info: >> generation finished with success in 4.5s !
info: >> 1 file(s) generated

# 查看生成的book.epub文件
$ tree -L 1
.
├── _book
├── book.epub
├── book.json
├── images
├── node_modules
├── README.md
└── SUMMARY.md
```

效果预览

Image 5.1.3.1 - ePub格式文件效果图预览

导出书籍为Mobi格式文件

Mobi是什么文件格式?

`mobi` 格式是一种广泛流行于网络的电子书格式，是亚马逊电子书格式，可以用亚马逊电子设备打开观看，也可用电脑打开，但需要使用特定的工具才能打开。

在书籍目录终端中执行 `gitbook mobi` 命令就可以生成一个名称为 `book.mobi` 的文件。

使用示例:

1. 插入logo

```
# 将书籍导出到Mobi文件，如果需要查看生成日志过程，可以加上--log=debug
$ gitbook mobi
info: 28 plugins are installed
info: 22 explicitly listed
info: loading plugin "insert-logo"... OK
info: loading plugin "favicon"... OK
info: loading plugin "search-pro"... OK
info: loading plugin "splitter"... OK
info: loading plugin "github"... OK
info: loading plugin "sharing-plus"... OK
info: loading plugin "code"... OK
info: loading plugin "advanced-emoji"... OK
info: loading plugin "emphasize"... OK
info: loading plugin "image-captions"... OK
info: loading plugin "anchor-navigation-expand"... OK
info: loading plugin "alerts"... OK
info: loading plugin "flexible-alerts"... OK
info: loading plugin "auto-scroll-table"... OK
info: loading plugin "lightbox"... OK
info: loading plugin "tbfed-pagefooter"... OK
info: loading plugin "hide-element"... OK
info: loading plugin "prism"... OK
info: loading plugin "chapter-fold"... OK
info: loading plugin "donate"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 38 pages
info: found 31 asset files
warn: "options" property is deprecated, use config.get(key) instead
warn: "options.generator" property is deprecated, use "output.name" instead
warn: "this.generator" property is deprecated, use "this.output.name" instead
warn: "navigation" property is deprecated
warn: "book" property is deprecated, use "this" directly instead
info: >> generation finished with success in 6.0s !
info: >> 1 file(s) generated

# 查看生成的book.mobi文件
$ tree -L 1
.
├── _book
├── book.json
├── book.mobi
├── node_modules
├── README.md
└── SUMMARY.md
```

预览效果

Image 5.1.4.1 - Mobi格式文件效果图预览

1. 编辑器类型

`Gitbook` 是基于 `Markdown` 语法文档的书籍制作工具，除了配置文件，所有书籍的页面文件都是 Markdown 类型的文档文件，目前可用的 **Markdown 的编辑器** 有很多种，简单向大家介绍一些比较常用的相关的 Markdown 编辑器。

> [!NOTE|style:flat] 按照 Markdown 编辑器的使用环境，可以将它们归纳为三类。
>
> 1. 平台集成工具：各大在线博客、社区平台自带的写作工具，比如CSDN、博客园、简书等。
> 2. 独立软件类：下载到自己机器上使用的独立产品，可以编辑本地文件，比如Mou、MarkdownEditor、Haroopad等。
> 3. 插件类：他自己本身是不能独立使用的，可以在你现有的主流编辑器安装，使你现有的编辑器具有Markdown的功能，比如Atom、WebStorm、Sublime Text等。

这三类软件分别面向三类不同 Markdown 需求的用户，大家可以根据自己的需求来选择使用。

2. MarkdownPad

`MarkdownPad` 被很多人称赞为windows下最好用的Markdown编辑器之一，不过仅支持windows。它有免费版和收费版（MarkdownPad Pro），一般情况下免费版就够用了，需要用pro版的可以自行购买。

MarkdownPad支持键盘快捷键和工具栏操作，即可添加标记也可移除，支持即时HTML预览、支持自定义配色方案、字体、大小和布局、支持音乐视频，可以导出HTML 和 PDF。

详细说明和使用参考官网：http://markdownpad.com

界面预览：

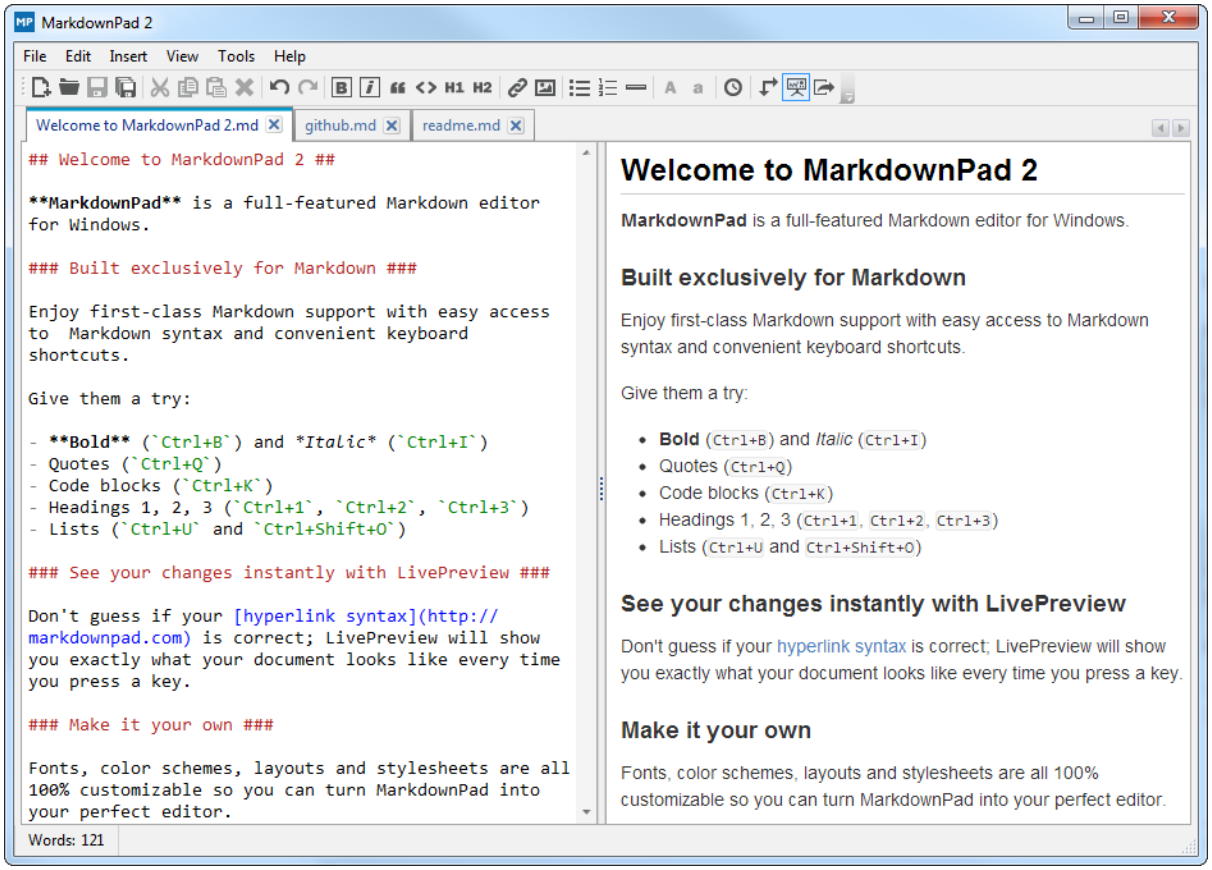

Image 6.1.1 - Markdownpad编辑器

3. Typora

`Typora` 也是非常用名，非常好用的 Markdown 编辑器，它的设计理念很不一样，是真正的即时预览型编辑器，不同于左右两个窗口的编辑器。Typora的设计理念就是极致简洁，它将「写字」和「预览」这两件事情合并了。

如果要修改已经写好的 Markdown 标记可以点击切换到“源代码模式”。

Typora 同样支持 Windows、OS X 和 Linux多个操作系统，支持数学编辑，可与 Word 直接格式转换，可以进行多种文档格式转换。Typora 流畅度和反应速度很快，特别适合那些手速快的人。

详细说明和使用参考官网：https://www.typora.io

界面预览：

Image 6.1.2 - Typora编辑器

4. Mou

1. 插入logo

`Mou` 是一款由国人开发的 Markdown 编辑器，支持实时预览，但是仅支持 苹果操作系统，可以说是目前最好用的免费 Markdown 编辑器，对汉字兼容性非常好。提供语法高亮、在线预览、同步滚动、全屏模式，支持自定保存、自动匹配，允许自定义主题等等。支持 CSS，HTML 和 PDF 导出等功能。

详细说明和使用参考官网：http://25.io/mou

界面预览：

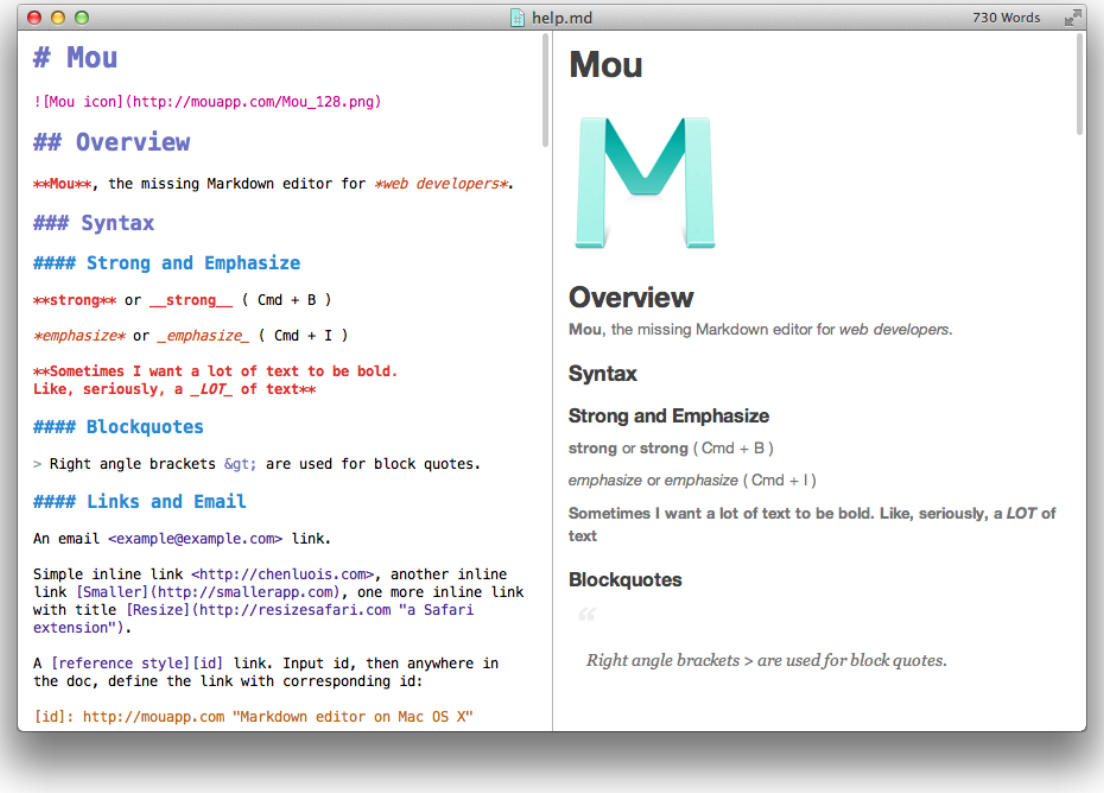

Image 6.1.3 - Mou编辑器

5. Atom

`Atom` 可以说是专门为程序员推出的一个文本编辑器，界面简洁，支持实时预览。功能非常多，除了Markdown 同时支持 CSS，HTML，JavaScript 等网页编程语言，还支持宏定义，自动分屏功能等。Atom还具有语义输入模式，比例输入code即会自动开启代码模式。

Atom 支持 windows、苹果、linux 等多种操作系统。Atom是由著名的 Github 平台出品的。

Atom有独立的软件，也支持插件方式。

详细说明和使用参考官网：https://atom.io

界面预览：

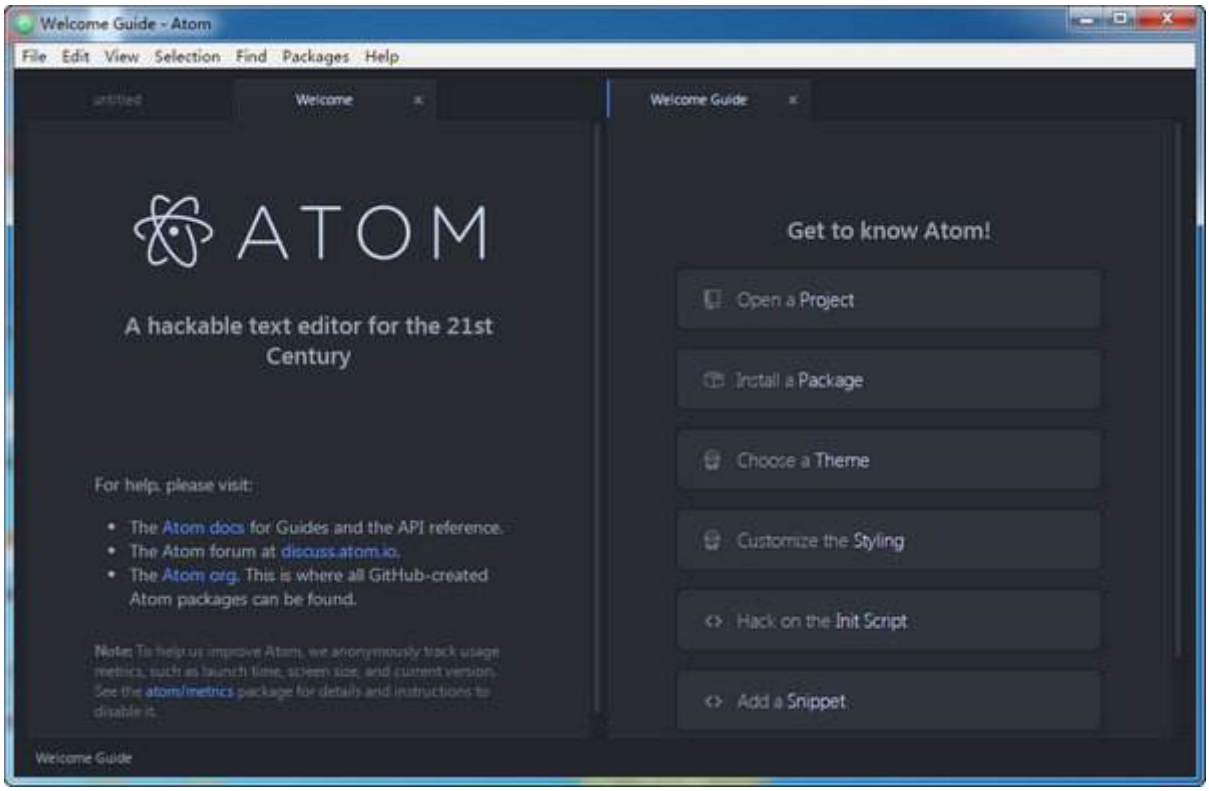

Image 6.1.4 - Atom编辑器

6. Haroopad

`Haroopad` 也是一款非常流行的编辑器，来自韩国。Haroopad 支持 Windows、Mac OS X 和 Linux三大操作系统。Haroopad 的特色是主题样式丰富，语法高亮支持无数种编程语言，几乎你能想到的编程语言他都支持。Ubuntu/Linux 用户使用该工具比例很高，Haroopad 也是开源免费的。Haroopad也支持导出HTML、PDF，也支持数学公式和流程图。

详细说明和使用参考官网：http://pad.haroopress.com/user.html

1. 插入logo

界面预览：

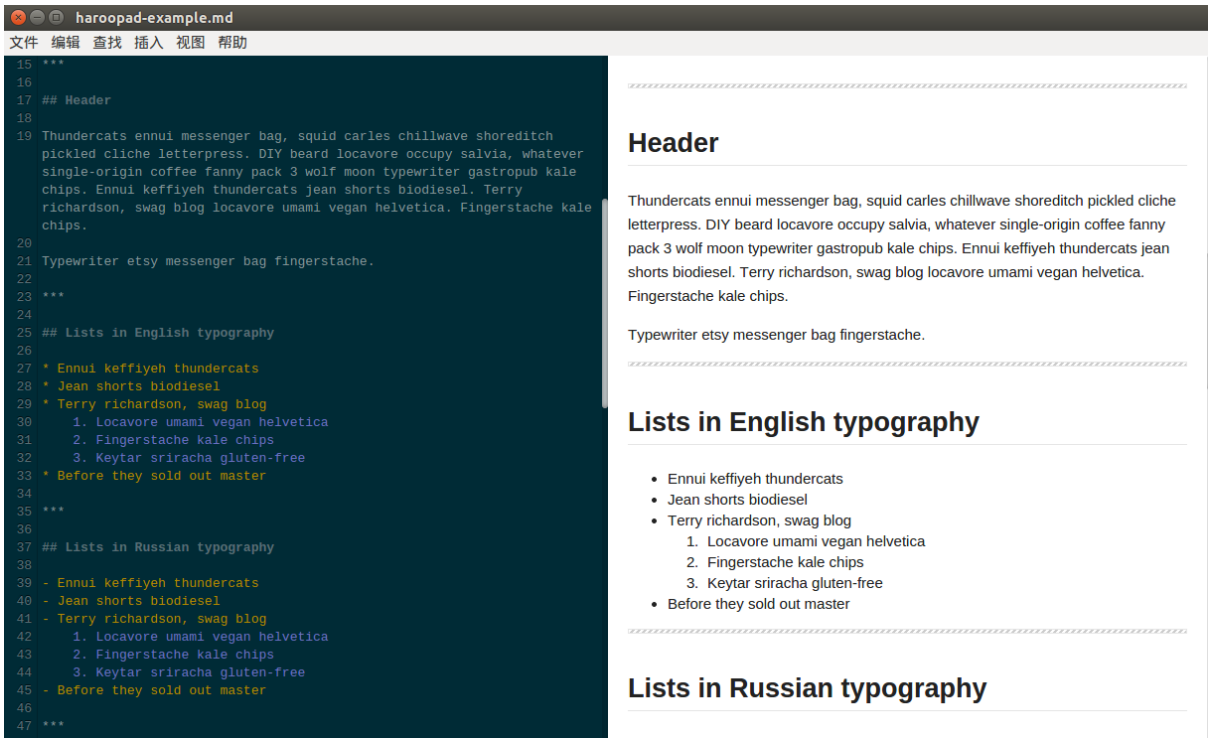

Image 6.1.5 - Haroopad编辑器

7. Cmd Markdown

`Cmd Markdown` 是一款不错的写作工具，同时也兼顾博客等写作平台，国内作业部落出品，同时支持Windows、苹果、Linux操作系统，也有 Web 在线创作平台，界面很舒服。

Cmd Markdown有独立的软件、也有平台集成版本。

详细说明和使用参考官网：https://www.zybuluo.com/mdeditor

界面预览：

Image 6.1.6 - Cmd Markdown编辑器

8. 最后

Markdown 的编辑器远远不止这些，这里就简单的介绍了一些比较常用的几款 Markdown 的文本编辑器，相信对于大多数人来说已经够用了。如果有其他编辑器功能需要的，可以根据自己的需要自行在网上找适合自己的编辑器。

发布书籍

在我们编写完成书籍后，可以将其编译的相关书籍的电子书文件发布给别人共享，目前主流的方式：

1. 将编译的 HTML 网站文件部署到自己的互联网中的 web 服务器中，供大家访问浏览，这样大家就可以实现公网共享访问浏览
2. 将编译的 HTML 网站文件部署到局域网下的web服务器，供有限的局域网内人员共享访问浏览，如公司的资料文档共享
3. 将编译的 HTML 网站文件部署到互联网中的代管服务器，比如：Github pages服务， Gitee pages服务等等，同样可以实现公网共享访问浏览
4. 将编译的 HTML、PDF、ePub、Mobi等文件以文件的方式共享给别人

当然分享的方式可能有很多其他的方式，但是主要还是网络预览的方式和文件的方式两种主要分享类型，接下来，笔者针对这两种方式简单的分享一下如何快速方便的使用互联网的方式来共享自己的书籍。

1. 通过相关互联网公司提供的相关静态网页服务，部署自己的数据，典型的有Github，Gitee 等相关 Pages 的服务。
2. 通过文件提供下载的方式来分享书籍。

发布到Github Pages

这个功能主要是将我们的书籍项目**编译成静态网站文件**，然后将静态网站文件上传到 `Github` 上，使用 Github 的 `Pages` 服务，创建一个属于自己的静态网站的一个功能，然后就可以利用 Github 来访问我们书籍了。

Github 官网也有对提供的 Pages有详细的说明，官网参考：https://pages.github.com

申请 Github 账号

首先的条件就是需要一个自己的 `Github` 账号，如果没有的话，可以去官网上申请注册一个自己的 Github 账号，官网地址：https://github.com 。

注册如下所示：

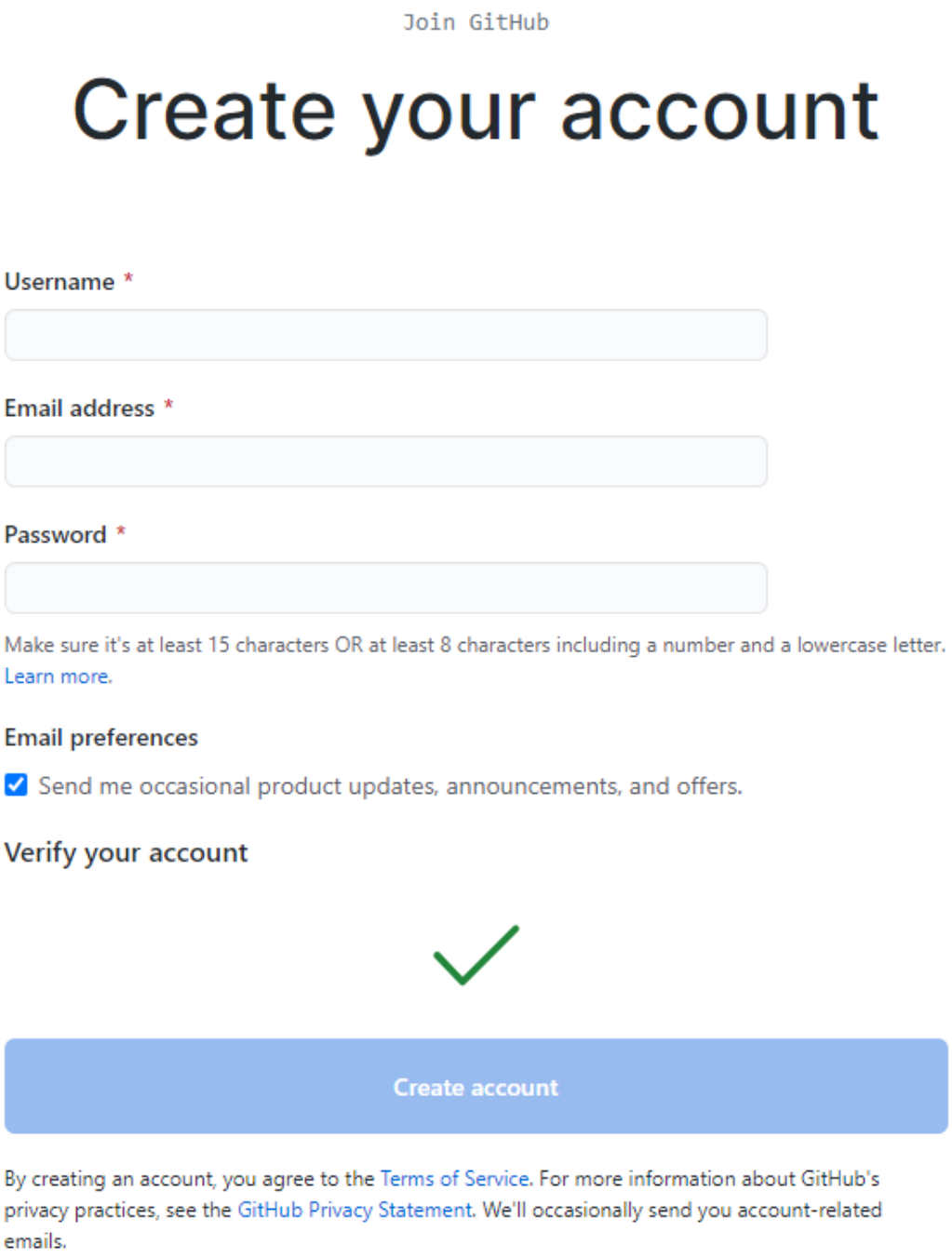

Image 7.1.1.1 - 注册Github账号

创建一个项目仓库

在 `Github` 中创建一个自己的项目仓库，名字按照你的想法去取，没有限制，当然建议大家尽量取一个有意义，简单易于理解的名字。

如下所示：

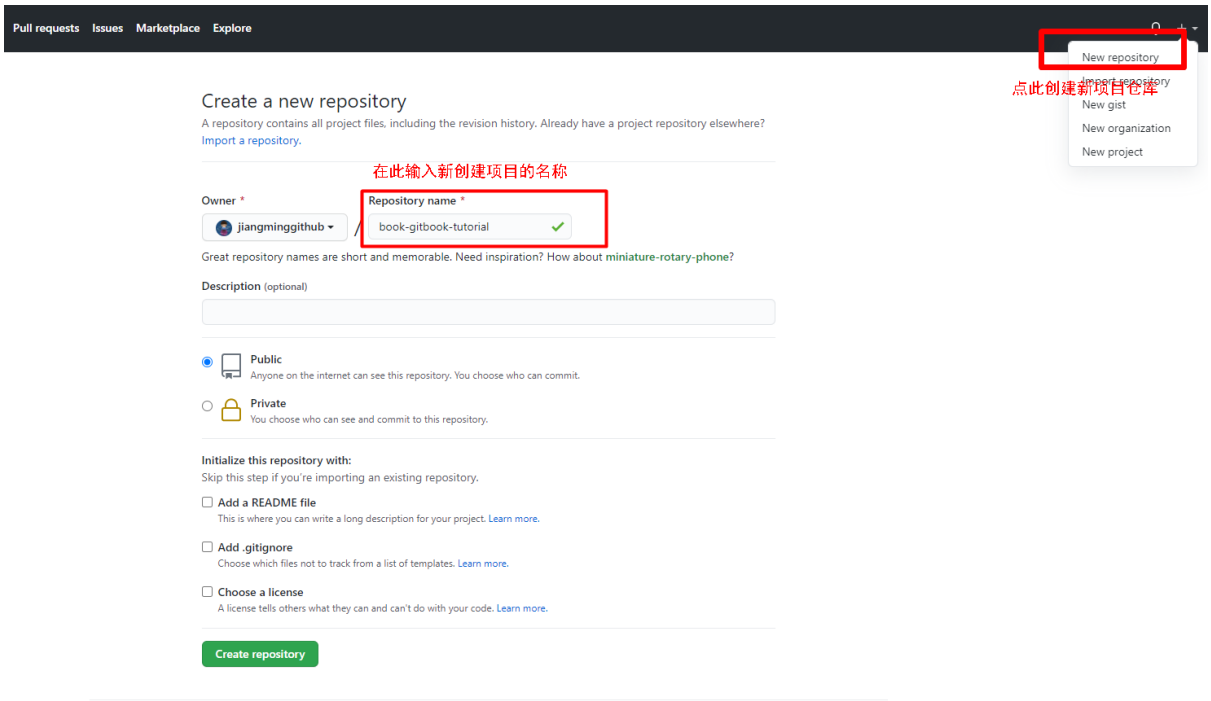

Image 7.1.1.2 - Github中创建一个仓库

推送静态网站文件到Github仓库中

将自己书籍项目编译出来的静态网站文件推送到刚刚创建的 `Github` 中的 `项目` 中。具体推送的方法这里就不具体细细的介绍了，可以参考相关的 Git 的教程和 Github 新创建项目中的使用说明。

> [!NOTE|style:flat]在这里推荐一个大家学习 `Git` 和 远程仓库的相关教程的学习网站，廖雪峰的Git教程：https://www.liaoxuefeng.com/wiki/896043488029600

github快速使用指南示例参考：

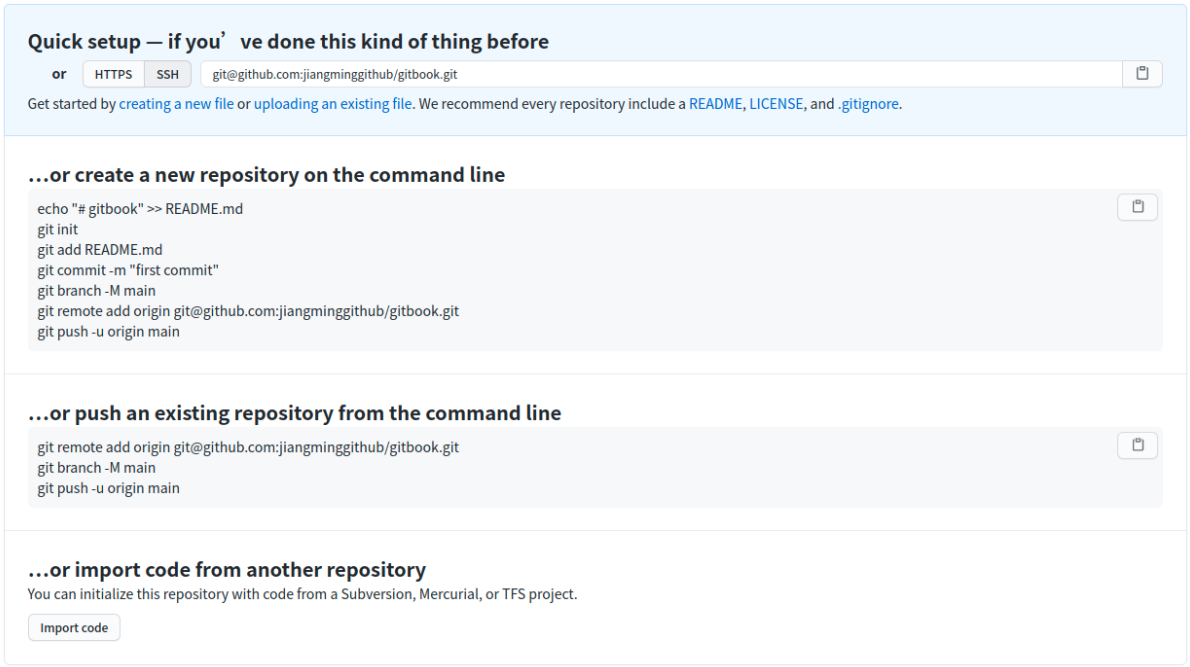

Image 7.1.1.3 - Github远程仓库快速指南

创建 Pages 服务并部署自己的静态网站

在需要创建 Pages 服务的项目中点击 “Settings” 按钮，滑动选项到 `Pages` 选项，选择需要创建 pages 服务的 `分支` 和 `部署的目标资源目录` （目前github仅支持指定 `项目的根目录` 和 `根目录下的docs目录` ），选择好之后保存就可以根据提示的 `网址链接` 来访问你部署的书籍的静态网站了。

> [!TIP|style:flat]
>
> 需要注意，目标仓库中必须有 index.html 才可以正常访问！

示意图：

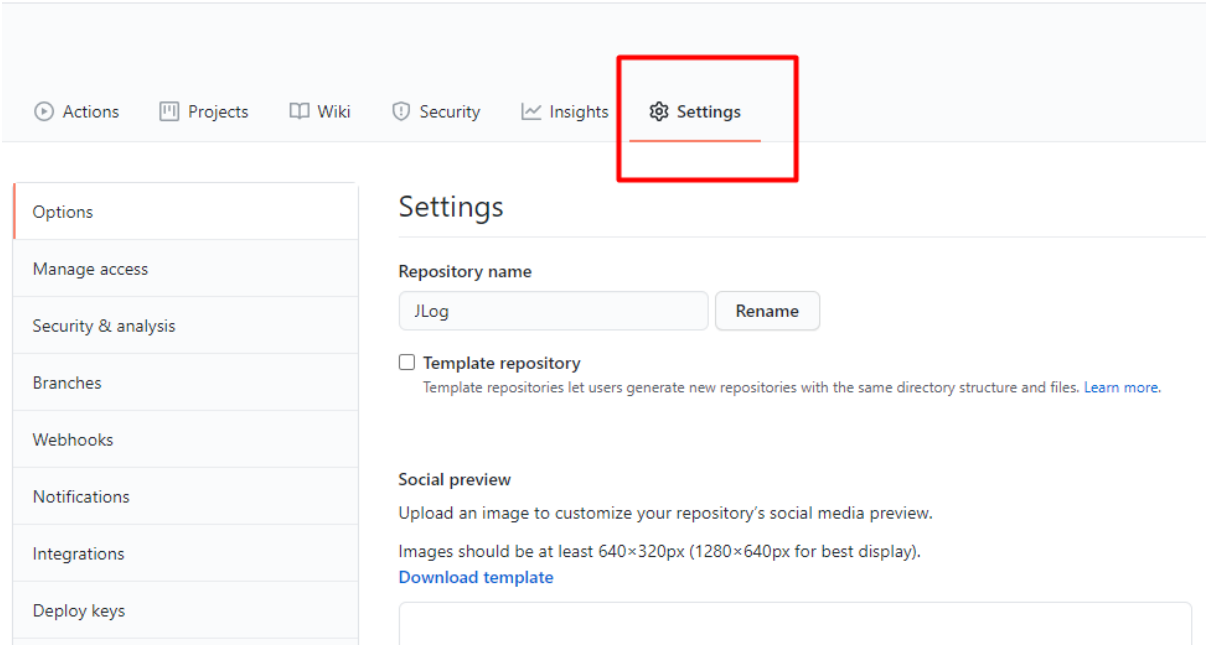

Image 7.1.1.4 - 点击Settings按钮

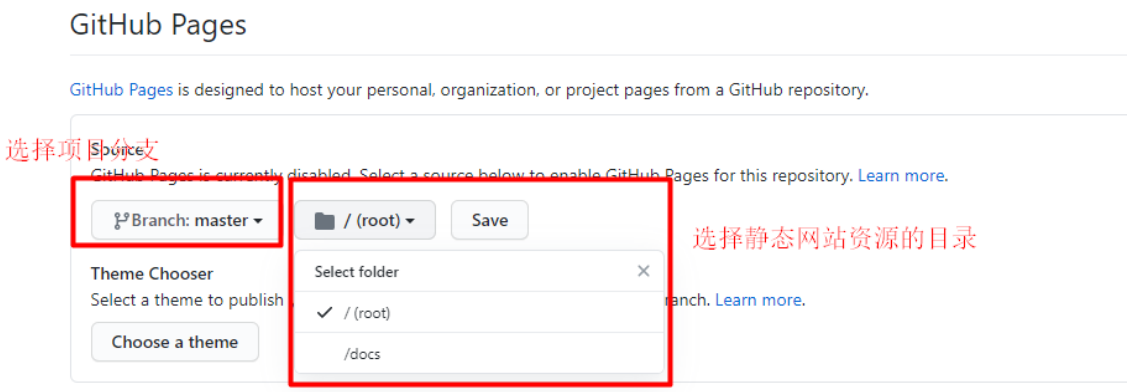

Image 7.1.1.5 - Pages配置

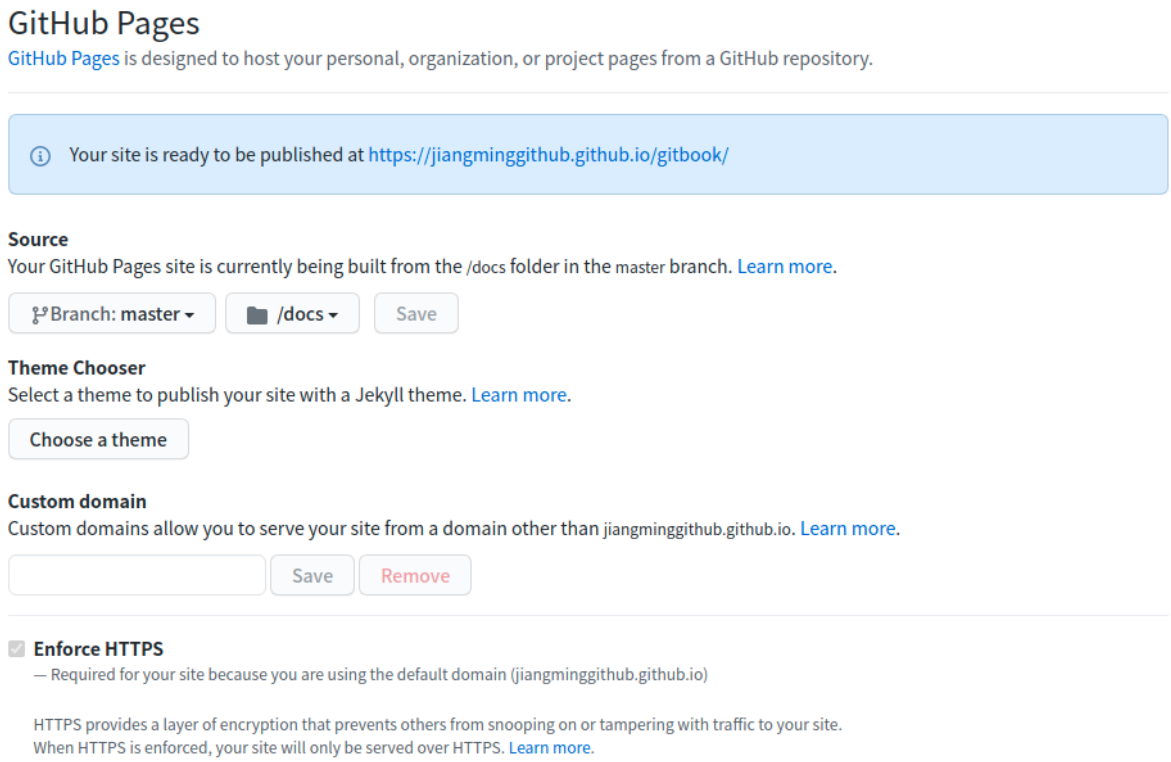

Image 7.1.1.6 - Pages设置完成

Image 7.1.1.7 - Pages在线网站访问

在线访问本书的github pages网站： https://jiangminggithub.github.io/gitbook/

发布到Gitee Pages

`Gitee` 是一个国内的 git 远程管理仓库，功能基本和 `Github` 一样，不过因为服务器在国内，所以访问速度比 Github 要快很多，同样这个功能主要是将我们的书籍项目**编译成静态网站文件**，然后将静态网站文件上传到 `Gitee` 上，使用 Gitee的 `Pages` 服务，创建一个属于自己的静态网站的一个功能，然后就可以利用 Gitee 来访问我们书籍了。

Gitee 官网也有对提供的 Pages有详细的说明，官网参考：
https://gitee.com/help/articles/4136#article-header0

申请 Gitee 账号

首先的条件就是需要一个自己的 `Gitee` 账号，如果没有的话，可以去官网上申请注册一个自己的 Gitee 账号，官网地址：https://gitee.com 。

注册示意图：

Image 7.1.2.1 - Gitee账号注册

创建一个项目仓库

在 `Gitee` 中创建一个自己的项目仓库，名字按照你的想法去取，没有限制，当然建议大家尽量取一个有意义，简单易于理解的名字。

示例图：

Image 7.1.2.2 - Gitee中创建一个仓库

推送静态网站文件到Github仓库中

将自己书籍项目编译出来的静态网站文件推送到刚刚创建的 `Gitee` 中的 `项目` 中。具体推送的方法这里就不具体细细的介绍了，可以参考相关的 Git 的教程和 Gitee 新创建项目中的使用说明。

> [!NOTE|style:flat]在这里推荐一个大家学习 `Git` 和 远程仓库的相关教程的学习网站，廖雪峰的Git教程：https://www.liaoxuefeng.com/wiki/896043488029600

gitee快速使用指南示例参考：

Image 7.1.2.3 - Gitee远程仓库快速指南

创建 Pages 服务并部署自己的静态网站

在需要创建 Pages 服务的项目中点击 “服务按钮” 按钮，选择 `Gitee Pages` 选项，选择需要创建 pages 服务的 `分支` 和 `部署的目标资源目录` ，选择好之后启动就可以根据提示的 `网址链接` 来访问你部署的书籍的静态网站了。

> [!TIP|style:flat]
>
> 需要注意，目标仓库中必须有 index.html 才可以正常访问！

示意图：

1. 插入logo

Image 7.1.2.4 - 点击服务按钮

Image 7.1.2.5 - Pages配置

Image 7.1.2.6 - Pages设置完成

Image 7.1.2.7 - Pages在线网站访问

在线访问本书的gitee pages网站： https://jiangming_gitee.gitee.io/gitbook/

1. 插入logo

发布电子书文件

将自己的 `Gitbook` 书籍项目导出各种电子书格式的文件单独发布出来的方式。可以通过 `网盘` 、 `邮件` 、 `微信` 、 `ftp` 等等很多方式来提供文件的共享。在此就不一一介绍怎么使用了，相信很多人可能已经很熟悉了。如果不熟悉的可以先学习一下互联网方面的相关知识，或者找一下相关的教程~

> [!TIP|style:flat]如何导出各种电子书格式的文件，请参考本书的 `书籍导出` 部分的章节介绍。

在线电子书：

在线访问本书的github pages网站： https://jiangminggithub.github.io/gitbook/

在线访问本书的gitee pages网站： https://jiangming_gitee.gitee.io/gitbook/

下载本书的电子书文件：

HTML 网站文件下载：点此下载

PDF 文件下载：点此下载

ePub 文件下载：点此下载

Mobi 文件下载：点此下载

结束语

不知不觉，本书的编写也到了尾声了，相信读完这本书后，你已经可以很好的去使用 `Gitbook` 来进行书籍的创作了。

碎碎念

> [!NOTE|style:flat] 在日常生活或者工作中，经常会学到一些知识或者遇到一些知识，都是需要什么学习什么，用什么查什么，没有形成一个系统的知识体系，在时间的流逝下，很多知识都会随之忘记了，所以可以通过书写记录笔记的方式来记录、回顾和总结的良好习惯。

学习是永无止境的，学的越多，才会发现自己的很多不足之处。正如站得高才能看得远，井底之蛙的眼光只有目光所到之处。希望大家也能在各自的领域，尽心耕耘，一份努力一份收货，多多学习，多多记录自己学到的知识，并能够很好的总结，不断提升自己的能力。

好了，碎碎念就到此结束 ~

回顾

与此同时，我们一起来简单回顾一下本书吧。本书主要分 `简单介绍` 、 `安装使用` 、 `结构配置` 、 `插件使用` 、 `书籍导出` 、 `编辑工具` ， `发布书籍` 等主要的几个模块来介绍如何使用 Gitbook 相关方面的知识。比较全面的介绍了常用的基础知识，可能也有不足之处或者错误的地方，欢迎大家补充 Gitbook 相关方便的知识，同样如果本书有什么不足或者有误的地方，也欢迎大家及时反馈，我将及时的反馈大家的问题，笔者在此将感激不尽 ~~~

联系

简单介绍一下笔者本人的情况，笔者是一名从事软件研发的程序员，热爱编程开发，电子数码，运动！目前主要从事 `智能汽车软件` 领域的研发工作，如有相关的软件研发方面技术交流和沟通的，也欢迎大家发送邮件到笔者的邮箱来一起沟通交流。

Image 8.1.1 - 作者签名

> [!TIP|style:flat]联系邮箱：
>
> - **Gmail 邮箱**: jiangmingyx@gmail.com
> - **126 邮箱**：jiangmingyx@126.com