

GitBook + Typora + Git 编写电子文档

jianshu.com/p/02caaaaa97ef

112019.04.01 22:42:40 字数 2,079 阅读 5,347

前言

近期使用 **Typora** 编写文档的时候，写一些篇章比较短的文章还好。但是当篇章过长，标题目录过多的时候，那样就会导致阅读效果很差。

为了提升更加好的体验效果，我下一步就需要考虑使用 **Gitbook** 的方式来编写管理文章了。

搭建GitBook 环境

下载安装Typora 和 git 工具

因为使用 GitBook + Typora + Git 三件套来进行电子书管理，所以你还需安装 Typora（一个很棒的支持 macOS、Windows、Linux 的 Markdown 编辑工具）和 Git 版本管理工具。戳下面：

- Typora 下载地址：<https://typora.io/>
- Git 下载地址：<https://git-scm.com/downloads>


安装nodejs

因为 GitBook 是基于 Node.js，所以我们首先需要安装 Node.js（下载地址：<https://nodejs.org/en/download/>），找到对应平台的版本安装即可。

The screenshot shows the Node.js download page. At the top, there's a navigation bar with links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, NEWS, and FOUNDATION. Below this, the 'Downloads' section is highlighted. It states 'Latest LTS Version: 10.15.3 (includes npm 6.4.1)' and 'Download the Node.js source code or a pre-built installer for your platform, and start developing today.' There are two main tabs: 'LTS Recommended For Most Users' and 'Current Latest Features'. Under the 'LTS' tab, there are three options: 'Windows Installer' (node-v10.15.3-x64.msi), 'macOS Installer' (node-v10.15.3.pkg), and 'Source Code' (node-v10.15.3.tar.gz). Below these, there are links for 'Windows Installer (.msi)', 'Windows Binary (.zip)', 'macOS Installer (.pkg)', 'macOS Binary (.tar.gz)', 'Linux Binaries (x64)', 'Linux Binaries (ARM)', and 'Source Code'. A table shows the available binaries for different architectures: 32-bit and 64-bit for x64, and ARMv6, ARMv7, and ARMv8 for ARM. A red arrow points to the '64-bit' link under the 'Current' tab. At the bottom, there's a section for 'Additional Platforms' with a link to 'macOS Binaries' and a '64-bit' link.

Platform	Architecture	Download Link
Windows	32-bit	node-v10.15.3-x64.msi
	64-bit	node-v10.15.3-x64.msi
macOS	32-bit	node-v10.15.3.pkg
	64-bit	node-v10.15.3.pkg
Linux	ARMv6	node-v10.15.3-armv6.tar.gz
	ARMv7	node-v10.15.3-armv7.tar.gz
	ARMv8	node-v10.15.3-armv8.tar.gz

因为我的系统版本是 win10 64位，那么我就下载这个版本来进行安装。

SSD (D:) > 01 software > 09 nodejs				
名称	修改日期	类型	大小	
 node-v10.15.3-x64.msi	2019/4/1 11:16	Windows Install...	16,708 KB	

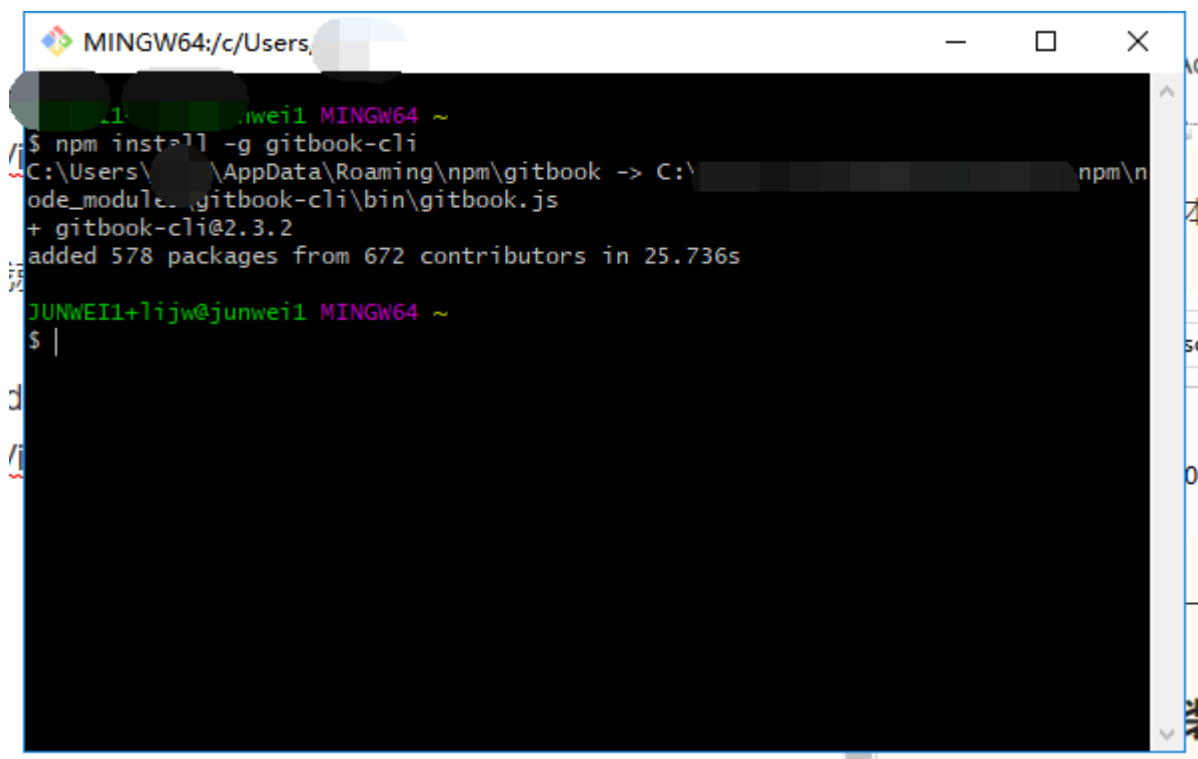
安装步骤只要下一步安装即可。

使用npm安装gitbook

现在安装 Node.js 都会默认安装 npm（node 包管理工具），所以我们不用单独安装 npm，打开命令行，执行以下命令安装 GitBook：

```
npm install -g gitbook-cli
```

使用**Git Bash**操作如下图：



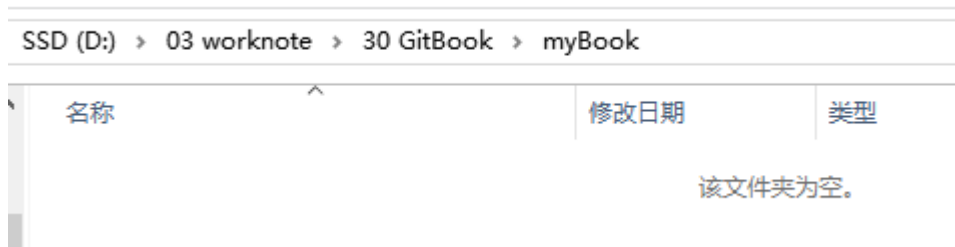
```
MINGW64:/c/Users/junwei1
$ npm install -g gitbook-cli
C:\Users\junwei1\AppData\Roaming\npm\gitbook -> C:\Users\junwei1\AppData\Local\npm\node_modules\gitbook-cli\bin\gitbook.js
+ gitbook-cli@2.3.2
added 578 packages from 672 contributors in 25.736s
JUNWEI@junwei1 MINGW64 ~
$ |
```

好了，准备好了三件套工具之后，就是来看看怎么编写**GitBook**了。

编写GitBook

创建电子书的文件夹目录

想象一下，现在你准备构建一本书籍，你在硬盘上新建了一个叫 mybook 的文件夹，按照以前的做法，你会新建一个 Word 文档，写上标题，然后开始巴滋巴滋地笔耕。



初始化Gitbook

但是现在有了 GitBook，你首先要做的是在 mybook 文件夹下执行以下命令：

使用 `gitbook init` 初始化gitbook，操作如下：

```
MINGW64:/d/03 worknote/30 GitBook/myBook
@0.7.1, depd@1.1.2, debug@2.2.0, mime@1.3.4, http-errors@1.3.1, on-finished@2.3.0
)
  resolve@1.1.7
  rmdir@1.2.0 (node.flow@1.2.3)
  gitbook-plugin-theme-default@1.0.7
  js-yaml@3.13.0 (esprima@4.0.1, argparse@1.0.10)
  fresh-require@1.0.3 (is-require@0.0.1, shallow-copy@0.0.1, sleuth@0.1.1, ast
w@1.3.0, through2@0.6.5, acorn@0.9.0, escodegen@1.11.1)
  tiny-lr@0.2.1 (parseurl@1.3.2, livereload-js@2.4.0, qs@5.1.0, debug@2.2.0, b
ody-parser@1.14.2, faye-websocket@0.10.0)
  cpr@1.1.1 (rimraf@2.4.5)
  gitbook-plugin-lunr@1.2.0 (html-entities@1.2.0, lunr@0.5.12)
  read-installed@4.0.3 (debuglog@1.0.1, util-extend@1.0.3, slide@1.1.6, readdi
r-scoped-modules@1.0.2, read-package-json@2.0.13)
  chokidar@1.5.0 (path-is-absolute@1.0.1, inherits@2.0.3, async-each@1.0.2, gl
ob-parent@2.0.0, is-binary-path@1.0.1, is-glob@2.0.1, anymatch@1.3.2, readdirp@2
.2.1)
  nunjucks@2.5.2 (asap@2.0.6, yargs@3.32.0, chokidar@1.7.0)
  gitbook-plugin-highlight@2.0.2 (highlight.js@9.2.0)
  moment@2.13.0
  gitbook-plugin-sharing@1.0.2 (lodash@3.10.1)
  i18n-t@1.0.1 (lodash@4.17.11)
  gitbook-markdown@1.3.2 (kramed-text-renderer@0.2.1, gitbook-html@1.3.3, kram
ed@0.5.6, lodash@4.17.11)
  cheerio@0.20.0 (entities@1.1.2, css-select@1.2.0, htmlparser2@3.8.3, jsdom@7
.2.2, lodash@4.17.11)
  gitbook-asciidoc@1.2.2 (gitbook-html@1.3.3, asciidoctor.js@1.5.5-1, lodash@4
.17.11)
  juice@2.0.0 (deep-extend@0.4.2, slick@1.12.2, batch@0.5.3, cssom@0.3.1, comm
ander@2.9.0, cross-spawn-async@2.2.5, web-resource-inliner@2.0.0)
  request@2.72.0 (tunnel-agent@0.4.3, aws-sign2@0.6.0, forever-agent@0.6.1, oa
uth-sign@0.8.2, is-typedarray@1.0.0, caseless@0.11.0, stringstream@0.0.6, aws4@1
.8.0, isstream@0.1.2, json-stringify-safe@5.0.1, tough-cookie@2.2.2, qs@6.1.2, n
ode-uuid@1.4.8, combined-stream@1.0.7, mime-types@2.1.22, bl@1.1.2, hawk@3.1.3,
http-signature@1.1.1, har-validator@2.0.6, form-data@1.0.1)
  npm@3.9.2
warn: no summary file in this book
info: create README.md
info: create SUMMARY.md
info: initialization is finished

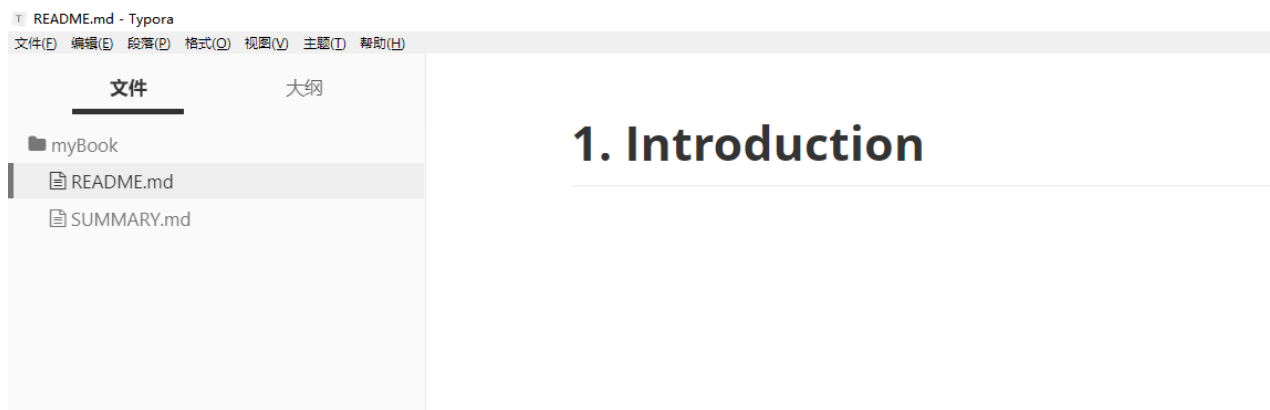
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ ls
README.md  SUMMARY.md
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$
```

执行完后，你会看到多了两个文件 —— README.md 和 SUMMARY.md，它们的作用如下：

- README.md —— 书籍的介绍写在这个文件里
- SUMMARY.md —— 书籍的目录结构在这里配置

使用Typora来编写框架内容

这时候，我们启动恭候多时的 Typora 来编辑这两个文件了：



编辑 SUMMARY.md 文件，内容修改为：

目录

- * [前言](README.md)
- * [第一章](Chapter1/README.md)
 - * [第1节：衣](Chapter1/衣.md)
 - * [第2节：食](Chapter1/食.md)
 - * [第3节：住](Chapter1/住.md)
 - * [第4节：行](Chapter1/行.md)
- * [第二章](Chapter2/README.md)
- * [第三章](Chapter3/README.md)
- * [第四章](Chapter4/README.md)

显示如下：



重新使用gitbook根据目录，初始化篇章

然后我们回到命令行，在 mybook 文件夹中再次执行 `gitbook init` 命令。GitBook 会查找 SUMMARY.md 文件中描述的目录和文件，如果没有则会将其创建。

```
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ ls
README.md  SUMMARY.md

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ ls
README.md  SUMMARY.md

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ gitbook init
info: create Chapter1/README.md
info: create Chapter1/衣.md
info: create Chapter1/食.md
info: create Chapter1/住.md
info: create Chapter1/行.md
info: create Chapter2/README.md
info: create Chapter3/README.md
info: create Chapter4/README.md
info: create SUMMARY.md
info: initialization is finished

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$
```

Typora 是所见即所得（实时渲染）的 Markdown 编辑器，这时候它是这样的：



启动服务，预览书籍

接着我们执行 `gitbook serve` 来预览这本书籍，执行命令后会对 Markdown 格式的文档进行转换，默认转换为 html 格式，最后提示“Serving book on http://localhost:4000”。
嗯，打开浏览器看一下吧：

```
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ gitbook serve
Live reload server started on port: 35729
Press CTRL+C to quit ...

info: 7 plugins are installed
info: loading plugin "livereload"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 9 pages
info: found 8 asset files
info: >> generation finished with success in 0.9s !

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



构建书籍

当你写得差不多，你可以执行 `gitbook build` 命令构建书籍，默认将生成的静态网站输出到 `_book` 目录。实际上，这一步也包含在 `gitbook serve` 里面，因为它们是 HTML，所以 GitBook 通过 Node.js 给你提供服务了。

```
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ ls
_book/  Chapter1/  Chapter2/  Chapter3/  Chapter4/  README.md  SUMMARY.md

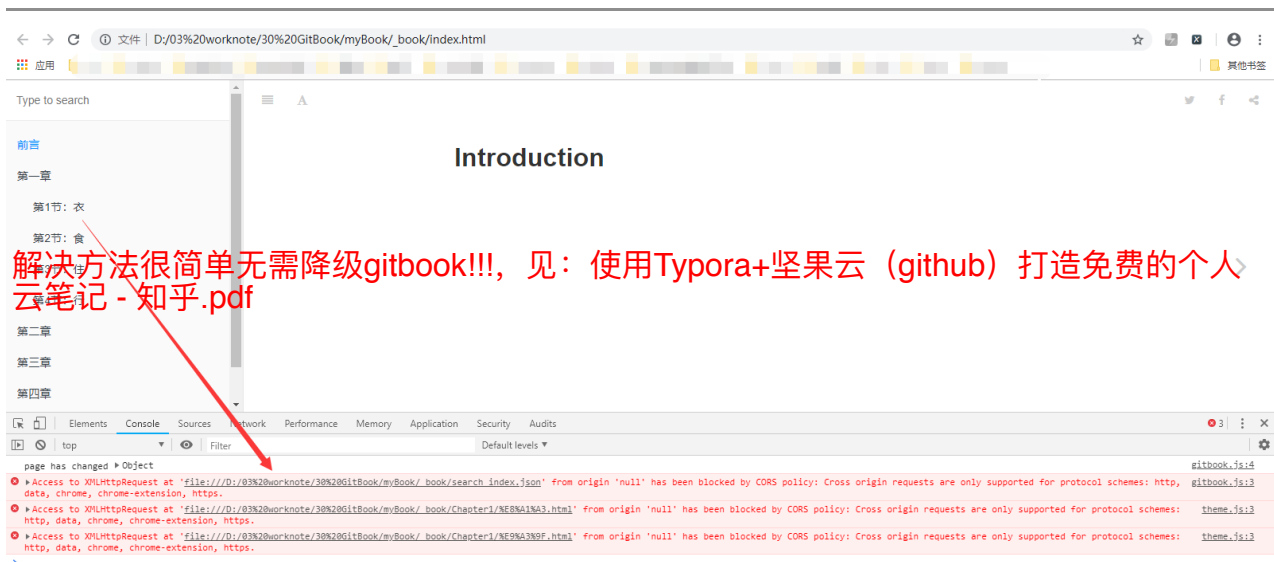
JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ gitbook build
info: 7 plugins are installed
info: 6 explicitly listed
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 9 pages
info: found 8 asset files
info: >> generation finished with success in 0.8s !

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ ls
_book/  Chapter1/  Chapter2/  Chapter3/  Chapter4/  README.md  SUMMARY.md

JUNWEI1+lijw@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$
```

D: > 03 worknote > 30 GitBook > myBook > _book >					搜索 "_book"
名称	修改日期	类型	大小		
Chapter1	2019/4/1 11:40	文件夹			
Chapter2	2019/4/1 11:40	文件夹			
Chapter3	2019/4/1 11:40	文件夹			
Chapter4	2019/4/1 11:40	文件夹			
gitbook	2019/4/1 11:40	文件夹			
index.html	2019/4/1 11:40	Chrome HTML D...	9 KB		
search_index.json	2019/4/1 11:40	JSON 文件	3 KB		

存在问题：3.2.3版本生成的静态html无法正常跳转



查看一下gitbook的版本，如下：

```
JUNWEI1+liju@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ gitbook -V
CLI version: 2.3.2
GitBook version: 3.2.3
```

由于gitbook版本过高导致生成的html打开后无法跳转章节

~~gitbook降低版本到 2.6.7 就可以子~~

~~可以使用如下命令~~

~~gitbook build --gitbook=2.6.7~~

~~但是降级就会遇到错误：~~

~~Error loading version latest: Error: Cannot find module 'internal/util/types'~~

```
JUNWEI1+liju@junwei1 MINGW64 /d/03 worknote/30 GitBook/myBook
$ gitbook build --gitbook=2.6.7
Error loading version latest: Error: Cannot find module 'internal/util/types'
  at Function.Module._resolveFilename (internal/modules/cjs/loader.js:582:15)
  at Function.Module._load (internal/modules/cjs/loader.js:508:25)
```


那么对付这个错误，就需要将node版本降级

• linux & ubuntu 环境

1. 安装node管理 n

```
sudo npm install -g n
```

2. 降低版本，更新npm

```
sudo n 6
```

3. 安装npm

```
sudo npm install npm -g
```

• window 环境

◦ nvm管理node版本

```
nvm install 6.16.0
```

```
nvm use 6.16.0
```

◦ 因为切换了新的node环境需要重新安装gitbook-cli

```
npm install -g gitbook-cli
```

降级之后，运行gitbook2.6.7的命令即可。

```
gitbook build gitbook=2.6.7
```

待转换完成后，将npm版本（例如 11.11.1）切回来即可，以免影响其他模块

• linux & ubuntu

```
sudo n 11.11.1
```

• window

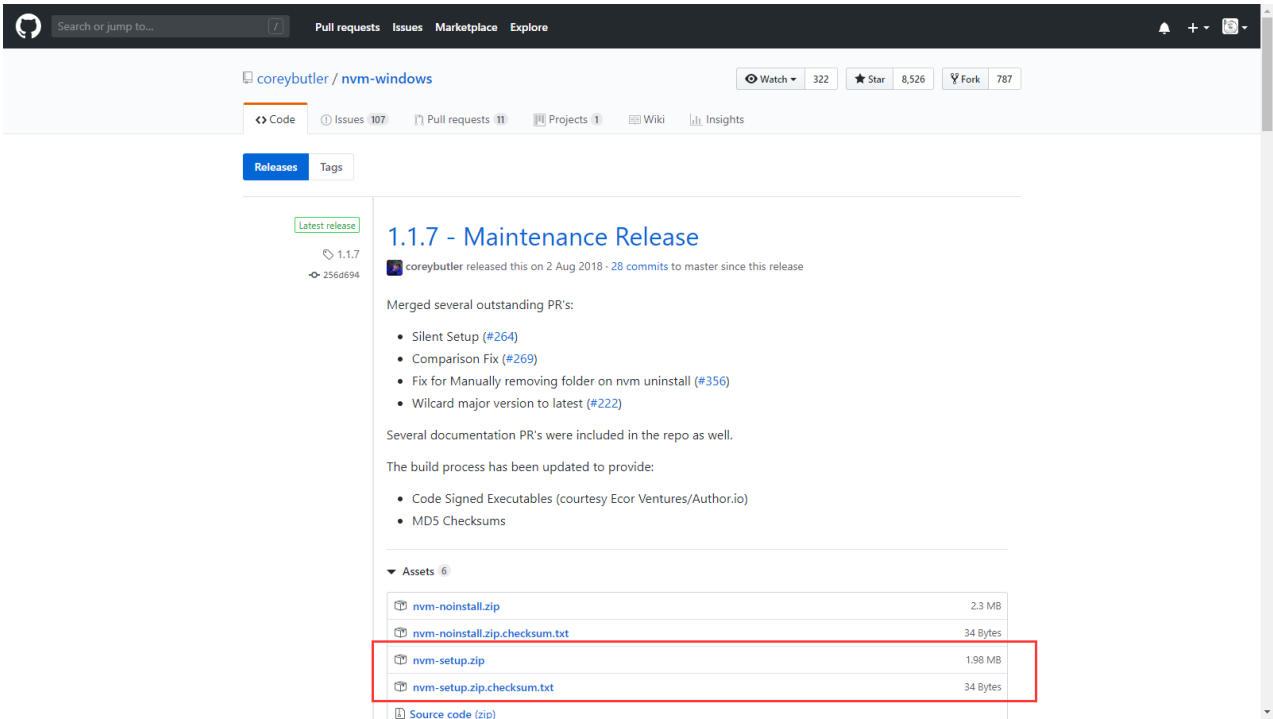
```
nvm use 你的最新版
```

因为我的环境是win10，那么就以win10的环境演示一遍。

在win10下安装nvm




Node版本管理器--nvm，可以运行在多种操作系统上。nvm for windows 是使用go语言编写的软件。我电脑使用的是Windows操作系统，所以我要记录下在此操作系统上nvm的安装和使用。

nvm最新的下载地址



~~我选择的是这个安装的版本，比较方便。前面的另一个是绿色版，需要配置。~~

SSD (D:) > 01 software > 09 nodejs >

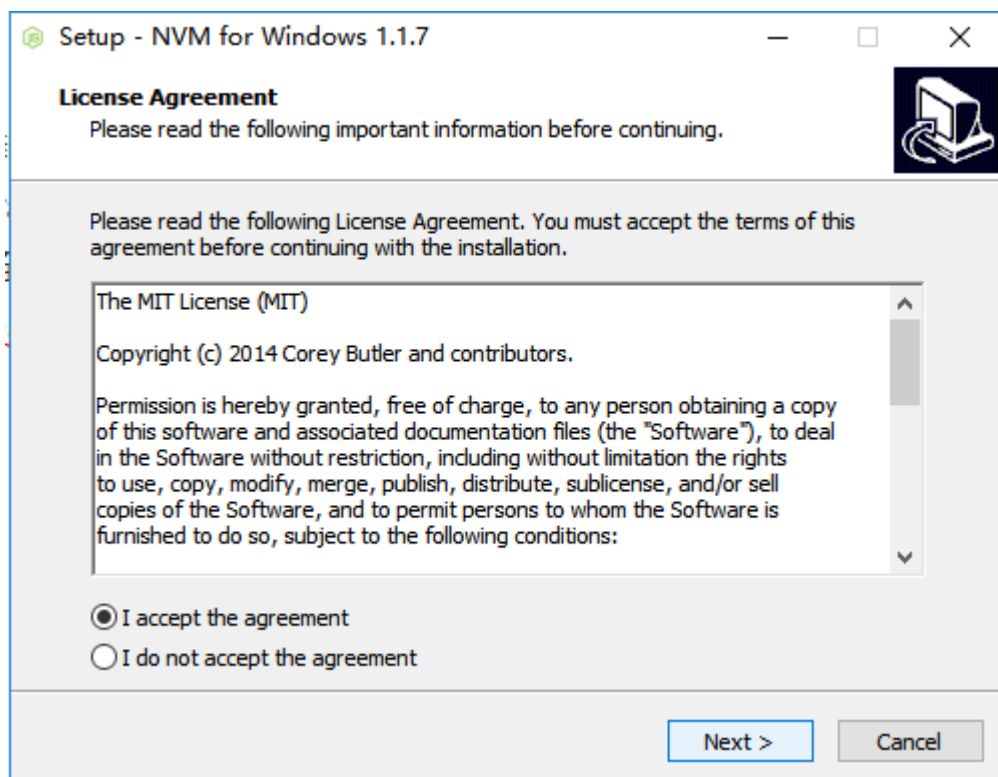
名称	修改日期	类型	大小
 node-v10.15.3-x64.msi	2019/4/1 11:16	Windows Install...	16,708 KB
 nvm-setup.exe	2018/8/8 13:46	应用程序	2,069 KB
 nvm-setup.zip	2019/4/1 14:16	WinRAR ZIP 压缩...	2,031 KB

~~注意事项:~~

~~安装之前的操作请注意: 在安装nvm for windows之前, 你需要卸载任何现有版本的node.js。并且需要删除现有的nodejs安装目录 (例如: "C:\Program Files\nodejs")。因为, nvm生成的symlink (符号链接/超链接)不会覆盖现有的 (甚至是空的) 安装目录。~~

~~你还需要删除现有的npm安装位置 (例如 "C:\Users\weiqin\AppData\Roaming\npm"), 以便正确使用nvm安装位置。~~

~~以上操作完成之后, 双击执行下载的setup文件。~~



~~Next之后，选择同意协议，之后选择nvm的本地安装目录，这里注意，nvm的安装路径名称中最好不要有空格。（注意：如果路径有空格，nvm use nodejs版本的时候就会报错。建议采用 D:\nvm 这样干净的路径。）~~

~~安装完毕后，在 cmd 输入 nvm ，会有相关的命令提示如下：~~

```
$ nvm
```

Running version 1.1.7.

Usage:

```

nvm arch                : Show if node is running in 32 or 64 bit mode.
nvm install <version> [arch] : The version can be a node.js version or "latest"
for the latest stable version.
                                Optionally specify whether to install the 32 or
64 bit version (defaults to system arch).
                                Set [arch] to "all" to install 32 AND 64 bit
versions.
                                Add --insecure to the end of this command to
bypass SSL validation of the remote download server.
nvm list [available]      : List the node.js installations. Type "available"
at the end to see what can be installed. Aliased as ls.
nvm on                   : Enable node.js version management.
nvm off                  : Disable node.js version management.
nvm proxy [url]          : Set a proxy to use for downloads. Leave [url]
blank to see the current proxy.
                                Set [url] to "none" to remove the proxy.
nvm node_mirror [url]    : Set the node mirror. Defaults to
https://nodejs.org/dist/. Leave [url] blank to use default url.
nvm npm_mirror [url]     : Set the npm mirror. Defaults to
https://github.com/npm/cli/archive/. Leave [url] blank to default url.
nvm uninstall <version>  : The version must be a specific version.
nvm use [version] [arch] : Switch to use the specified version. Optionally
specify 32/64bit architecture.
                                nvm use <arch> will continue using the selected
version, but switch to 32/64 bit mode.
nvm root [path]          : Set the directory where nvm should store
different versions of node.js.
                                If <path> is not set, the current root will be
displayed.
nvm version              : Displays the current running version of nvm for
Windows. Aliased as v.

```

如果要升级nvm

如果要升级的话，请重新下载最新的安装程序。并直接运行安装程序。它将安全的覆盖需要更新的文件，而无需关心nodejs的安装。

此次安装需要确保和上次使用相同的安装目录。

如果你最初安装到默认位置，则只需一直点击"下一步"，直到完成。

使用nvm

nvm for windows是一个命令行工具，在控制台输入nvm,就可以看到它的命令用法。基本命令有：

`nvm arch [32|64]` : 显示node是运行在32位还是64位模式。指定32或64来覆盖默认体系结构。

`nvm install <version> [arch]`: 该可以是node.js版本或最新稳定版本latest。(可选[arch])指定安装32位或64位版本(默认为系统arch)。设置[arch]为all以安装32和64位版本。在命令后面添加`--insecure` , 可以绕过远端下载服务器的SSL验证。

`nvm list [available]`: 列出已经安装的node.js版本。可选的available, 显示可下载版本的部分列表。这个命令可以简写为`nvm ls [available]`。

`nvm on`: 启用node.js版本管理。

`nvm off`: 禁用node.js版本管理(不卸载任何东西)

`nvm proxy [url]`: 设置用于下载的代理。留[url]空白, 以查看当前的代理。设置[url]为none删除代理。

`nvm node_mirror [url]`: 设置node镜像, 默认为<https://nodejs.org/dist/>。我建议设置为淘宝的镜像<https://npm.taobao.org/mirrors/node/>

`nvm npm_mirror [url]`: 设置npm镜像, 默认为<https://github.com/npm/npm/archive/>。我建议设置为淘宝的镜像<https://npm.taobao.org/mirrors/npm/>

`nvm uninstall <version>`: 卸载指定版本的nodejs。

`nvm use [version] [arch]`: 切换到使用指定的nodejs版本。可以指定32/64位[arch]。nvm use <arch>将继续使用所选版本, 但根据提供的值切换到32/64位模式的<arch>

`nvm root [path]`: 设置 nvm 存储node.js不同版本的目录 , 如果未设置, 将使用当前目录。

`nvm version`: 显示当前运行的nvm版本, 可以简写为`nvm v`

使用nvm安装nodejs

首先查看一下当前有哪些可用的版本, 如下:

```
$ nvm list available
```

CURRENT	LTS	OLD STABLE	OLD UNSTABLE
11.13.0	10.15.3	0.12.18	0.11.16
11.12.0	10.15.2	0.12.17	0.11.15
11.11.0	10.15.1	0.12.16	0.11.14
11.10.1	10.15.0	0.12.15	0.11.13
11.10.0	10.14.2	0.12.14	0.11.12
11.9.0	10.14.1	0.12.13	0.11.11
11.8.0	10.14.0	0.12.12	0.11.10
11.7.0	10.13.0	0.12.11	0.11.9
11.6.0	8.15.1	0.12.10	0.11.8
11.5.0	8.15.0	0.12.9	0.11.7
11.4.0	8.14.1	0.12.8	0.11.6
11.3.0	8.14.0	0.12.7	0.11.5
11.2.0	8.13.0	0.12.6	0.11.4
11.1.0	8.12.0	0.12.5	0.11.3
11.0.0	8.11.4	0.12.4	0.11.2
10.12.0	8.11.3	0.12.3	0.11.1
10.11.0	8.11.2	0.12.2	0.11.0
10.10.0	8.11.1	0.12.1	0.9.12
10.9.0	8.11.0	0.12.0	0.9.11
10.8.0	8.10.0	0.10.48	0.9.10

This is a partial list. For a complete list, visit <https://nodejs.org/download/release>

可以从清单中看出, nodejs的版本较多, 本次的目的是为了降级提供 [Gitbook](#) 导出的html 可以跳转。

那么下面就安装一个较低的版本, 如下:

```
nvm install 6.16.0
nvm use 6.16.0
```

再重新安装Gitbook

```
npm install -g gitbook-cli
```

再生成Gitbook

```
gitbook build --gitbook=2.6.7
```

打开生成html，侧边栏可以跳转章节了，但是没有了最新版本的search功能，如下：



不过，没关系了，如果要用齐全的功能，最好就是直接使用nodejs搭起服务，这样功能就完整了。

GitBook其他相关命令

当然，build 命令可以指定路径：

```
$ gitbook build [书籍路径] [输出路径]
```

serve 命令也可以指定端口：

```
$ gitbook serve --port 2333
```

你还可以生成 PDF 格式的电子书：

```
$ gitbook pdf ./ ./mybook.pdf
```

缺少ebook-convert,需要安装操作系统对应的calibre

生成 epub 格式的电子书：

```
$ gitbook epub ./ ./mybook.epub
```

安装该应用完成后，建立命令行链接sudo ln -s /Applications/calibre.app/Contents/MacOS/ebook-convert /usr/local/bin

生成 mobi 格式的电子书：

```
$ gitbook mobi ./ ./mybook.mobi
```

大功告成，找到一本gitbook的的github地址仓库，git clone到本地或者直接下载压缩包解压后进入该文件夹下，执行

如果生成不了，你可能还需要安装一些工具，比如 ebook-convert。或者在 Typora 中安装 Pandoc 进行导出。

除此之外，别忘了还可以用 Git 做版本管理呀！在 mybook 目录下执行 `git init` 初始化仓库，执行 `git remote add` 添加远程仓库（你得先在远端建好）。接着就可以愉快地 `commit`，`push`，`pull ...` 啦！

参考文献

GitBook 从懵逼到入门

GitBook基本使用

gitbook build 生成的HTML无法跳转问题

windows上NVM安装与使用

59人点赞 >

Typora系列

"你的赞赏是我创作最大的动力"

还没有人赞赏，支持一下

59赞60赞

赞赏