

---

# Table of Contents

前言	1.1
gitbook介绍	1.2
安装和设置	1.3
目录结构	1.4
菜单与页面	1.5
配置	1.6
词汇表	1.7
多语言	1.8
markdown语法	1.9
生成电子书和pdf	1.10
成功案例	1.11
gitbook的一些资源	1.12

# 前言

先说几个定义

## 知识

通过学习、实践或探索所获得的认识、判断或技能。

注1：知识可以是显性的，也可以是隐性的；可以是组织的，也可以是个人的。

注2：知识可包括事实知识、原理知识、技能知识和人际知识。

对知识可以有这样的概述：知识是人类通过实践对自然、社会和思维活动形态及其规律认识和描述的信息。他来源于人类实践，并反作用于实践，对实践具有指导性作用。知识是人类认识世界和改造世界的实践的产物。知识是信息的一部分，是对自然、社会、思维活动形态、规律的认识把握，并加以描述。描述是知识和认识的关键区别，凡是认识了的东西都可以描述，知识经过描述，就必须依赖物质载体才能存储起来，才能流传、积累、交流、发展、开发、利用。

## 知识管理

所谓知识管理的定义为，在组织中构建一个量化与质化的知识系统，让组织中的资讯与知识，透过获得、创造、分享、整合、记录、存取、更新、创新等过程，不断的回馈到知识系统内，形成永不间断的累积个人与组织的知识成为组织智慧的循环，在企业组织中成为管理与应用的智慧资本，有助于企业做出正确的决策，以适应市场的变迁。

一句话概括为：知识管理是对知识、知识创造过程和知识的应用进行规划和管理的活动。

常见的系统有：早期的blog、空间，现在更专业的：wiki

## 个人知识管理

个人知识管理是一种新的知识管理的理念和方法，能将个人拥有的各种资料、随手可得的信息变成更具价值的知识，最终利于自己的工作、学习和生活。通过对个人知识的管理，人们可以养成良好的学习习惯，增强信息素养，完善自己的专业知识体系，提高自己的能力和竞争力，为实现个人价值和可持续发展打下坚实基础。

传统意义上的知识管理都是企业做的，但是个人和企业是一对多的关系，而且有一些是和企业不相关的兴趣爱好之类的。

## 知识管理工具

印象笔记、有道云笔记、wiki、blog、gitbook等

各种工具都各有优缺点，gitbook相对是比较小众的，比较适合程序员的，比纯vim好的是不用手动去写各种标记。相当于是个简版的word。

---

# gitbook介绍

GitBook是一款文档编辑工具。它的功能类似金山WPS中的Word或者微软Office中的Word的文档编辑工具。它可以用来写文档、建表格、插图片、生成pdf。当然，以上的功能WPS、Office可能做得更好，但是，GitBook还有更强大的功能：它可以用文档建立一个网站，让更多人了解你写的书，另外，最核心的是，他支持Git，也就意味着，它是一个分布式的文档编辑工具。你可以随时随地来编写你的文档，也可以多人共同编写文档，哪怕多人编写同一页文档，它也能记录每个人的内容，然后告诉你他们之间的区别，也能记录你的每一次改动，你可以查看每一次的书写记录和变化，哪怕你将文档都删除了，它也能找回来！这就是它继承Git后的厉害之处！

## 分布式、多人协同的全新书写方式！

在GitBook中，你可以使用Markdown或者AsciiDoc语法，加上几个命令就能创建一本漂亮的图书。这种便利就像Node.js一样。GitBook支持GitHub或者git来管理文章的改动和版本。你可以参考这个例子：[ReduxJS documentation](#)或[WebMagic Java爬虫文档](#)。

你可以在自己电脑上使用GitBook来写一本书，也可以通过[GitBook.com](#)在线方式写一本书。还可以在自己电脑上安装[桌面版的GitBook编辑器](#)。

你可以加入[GitBook社区](#)也可以在推特账号[@GitBookIO](#)或脸谱账号[GitBook](#)中获取我们的最新进展或者留言。

完整的中文文档在[gitbook.hushuang.me](#)上。

## 开始

使用GitBook，不需要联网就可以在您的计算机上创建本地书籍，也可以通过GitBook.com网站在线编辑你的书，不用担心电脑故障导致文章丢失，因为它就在云端。想要试试，就开始看[安装说明文档](#)。

## 举个例子

GitBook可以用来写书，API文档、公共文档，企业手册，论文，研究报告等。这里有一堆[示例列表](#)我相信总有一款适合你。

## 帮助和支持

如果您在使用GitBook编写文档和书籍的时候，遇到问题，我们随时乐意帮助解决。您可以在[联系地址](#)上通过以下表单提出问题，或在[GitHub问题列表](#)中提出你的问题。

## 特性

- [Markdown or AsciiDoc 语法](#)
- 直接导出为网站或者[电子书 \(pdf, epub, mobi\)](#)
- [多语言](#)
- [目录、大纲](#)
- [封面](#)
- [模板和变量](#)
- [模板继承](#)
- [插件](#)
- [优美的主题](#)

## 发布你自己的书

你可以使用[GitBook.com](#)写一本关于自己的书，例如 "XXX自传"，可以设置成私人的或者公开的，写的过程中通过 git push 命令就更新到云端了。

## 授权许可

GitBook的授权许可使用的是Apache License, Version 2.0.

# 安装和使用GitBook

完成GitBook的安装只需要几分钟。

## GitBook.com

[GitBook.com](#)为您提供简单高效的图书在线撰写、发布和托管方案，你可以通过[GitBook.com](#)进行在线编辑、或者使用[GitBook 本地编辑器](#)在本地电脑上编辑。

## 本地安装

### 要求

GitBook的安装非常简单。您的系统只需满足这两个要求：

- NodeJS (推荐使用v4.0.0及以上版本)
- Windows, Linux, Unix或Mac OS X

### 使用NPM安装

安装GitBook的最好方法是通过 NPM 安装。在已经安装好NodeJS和NPM的电脑上，通过命令行窗口，输入以下命令安装GitBook：

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

gitbook-cli 是安装和管理GitBook版本库的程序。它会自动安装GitBook所需的模块来创建一本书。

### 创建一本书

GitBook通过以下命令在当前目录创建一本书：

```
$ gitbook init
```

如果你想用现有的目录来创建一本书，你可以通过运行 `gitbook init ./directory` 来实现

使用下面的命令预览您创建的图书：

```
$ gitbook serve
```

或者使用以下命令构建静态网站：

```
$ gitbook build
```

### 安装其他版本

`gitbook` 命令可以方便地下载和安装不同版本的GitBook来测试你的书：

```
$ gitbook fetch 4.0.0-alpha.1
```

使用 `gitbook ls-remote` 列出可用于安装的远程版本。

```
$ gitbook ls-remote
```

```
Available GitBook Versions:
```

```
4.0.0-alpha.5, ...部分省略..., 4.0.0-alpha.1, 3.2.2, 3.2.1, ...部分省略..., 2.0.0-alpha.1
```

```
Tags:
```

```
latest : 3.2.2
```

```
pre : 4.0.0-alpha.5
```

## 调试

您可以使用 `--log=debug` 和 `--debug` 来获得更详细的错误消息（堆栈跟踪）。例如：

```
$ gitbook build ./ --log=debug --debug
```

or

```
$ gitbook serve ./ --log=debug --debug
```

# 目录结构

GitBook使用[SUMMARY](#)文件管理目录结构，文件支持Markdown和Asciidoc两种语法。 GitBook按照[SUMMARY](#)文件中的目录结构生成HTML。如果你通过GitBook创建一本支持多语言文档，目录结构会稍微不同，具体可参考[多语言环境](#)。

一般GitBook目录如下：

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

简单介绍每一项：

文件	说明
book.json	保存 <a href="#">配置文件</a> 数据(可选)
README.md	简介 - 书籍的简单介绍(必填)
SUMMARY.md	目录(参见 <a href="#">目录管理</a> )(可选)
GLOSSARY.md	字段/注释 - 专业术语列表(参见 <a href="#">词汇表</a> )(可选)

## 静态文件

静态文件是在 `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，

## 忽略的文件和文件夹

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。这些文件中的格式遵循与 `.gitignore` 相同的约定：

```
# 井号代表这一行注释

# 忽略文件test.md
test.md

# 忽略目录“bin”中的所有内容
bin/*
```

## 以子目录的方式与项目集成

对于软件项目，可以使用子目录(如 `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：

```
.
├── book.json
└── docs/
```

```
|── README.md  
└── SUMMARY.md
```

book.json 中的配置如下：

```
{  
    "root": "./docs"  
}
```

# 菜单与页面

---

## 菜单

GitBook使用一个 `SUMMARY.md` 文件来定义文档的菜单。

`SUMMARY.md` 中 [] 内的内容是标题，() 内是文档的路径，章节和子章节用四个空格或者 `tab` 键来分级。

### 简单示例

```
# 概述
### 第一部分
*
[
    第一部分
](
    part1/README.md
)

    *
        [
            第二部分
        ](
            part2/README.md
        )

            *
                [
                    我们喜欢社交网络](part2/README.md#feedback)
                *
                    [更好的写作工具](part2/README.md#tools)
```

每一个章节都有一个专用的页面 ( `part1/README.md#` )，并被分割成子章节。

### 区域导航

文章可以使用区域导航定位到文件的特定部分。在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

```
# 概述
### 第一部分
*
[
    第一部分
](
    part1/README.md
)

    *
        [
            第二部分
        ](
            part2/README.md
        )

            *
                [
                    我们喜欢社交网络](part2/README.md#feedback)
                *
                    [更好的写作工具](part2/README.md#tools)
```

### 部分

内容表可以分为由标题或水平线分隔的部分：

```
# 章节标题

这里是一些简单的介绍。

## 第一节

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

## 第二节

...
```

部分只是章节组，没有对应的页面，但根据不同主题，它会显示在导航中。

## 页面

### Markdown语法

GitBook默认使用Markdown语法编写页面。 Markdown语法可参考[GitBook Markdown语法](#)章节。也可以选择[AsciiDoc语法](#)。

#### 章节文件的示例

```
# 本章标题

这里是介绍。

## 第1节

Markdown将决定
**书的结构**
## 第2节

...
```

### 顶部描述

页面可以用它作为描述。 它使用 YAML 格式的风格，在三条虚线之间。 文档中也可以不写顶部描述。

**特别提示:**在没有安装支持插件之前，不要在文件中使用，否则编译或者运行会失败。

这里有一个基本的例子：

```
---
description:
  This is a short description of my page
---
# The content of my page
```

...

顶部描述的内容可以定义自己的变量，可以参考[页面变量](#)，以便您可以在模板中使用它们。

# 配置

GitBook允许您使用灵活的配置自定义书籍和文档。这些选项在 `book.json` 文件中指定。对于不熟悉JSON语法的作者，您可以使用[JSONlint](#)等工具验证语法。

## 常规设置

变量	说明
<code>root</code>	包含所有图书文件的根文件夹的路径，除了 <code>book.json</code>
<code>structure</code>	指定自述，摘要，词汇表等的路径。请参见 <a href="#">结构段落</a> 。
<code>title</code>	书的标题，默认值从README中提取。在GitBook.com上，此字段已预填。
<code>description</code>	您的图书说明，默认值从自述文件中提取。在GitBook.com上，此字段已预填。
<code>author</code>	作者姓名。在GitBook.com上，此字段已预填。
<code>isbn</code>	书的国际码ISBN
<code>language</code>	<a href="#">语言ISO规范</a> 的书的语言，默认值是 <code>en</code>
<code>direction</code>	文本的方向。可以是 <code>rtl</code> 或 <code>ltr</code> ，默认值取决于 <code>language</code> 的值
<code>gitbook</code>	GitBook的版本。使用 <a href="#">SemVer</a> 规范并接受诸如“ <code>&gt; = 3.0.0</code> ”的条件

## 插件

插件及其配置在 `book.json` 中指定。有关更多详细信息，请参阅[插件部分](#)。

从3.0.0版本开始，GitBook可以使用主题。有关详细信息，请参阅[主题部分](#)。

变量	说明
<code>plugins</code>	要加载的插件列表
<code>pluginsConfig</code>	插件配置

## 结构体

除了 `root` 变量，你可以告诉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的根(或每个语言书的根)。不接受诸如 `dir/MY_README.md` 之类的路径。

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

## PDF选项

PDF输出可以使用 `book.json` 中的一组选项来定制：



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”, “b3” '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”)

# 词汇表

您可以指定要显示为注释的术语及其相应的定义。 基于这些术语，GitBook在编译的时候会自动构建索引并在页面中突出显示这些术语。

`GLOSSARY.md` 是一个 `h2` 标题的列表，以及一个描述段落：

```
## 期限  
此术语的定义  
  
## 其他术语  
它的定义，可以包含粗体文本  
所有其他种类的内联标记...
```

# 多语言

GitBook 支持多种语言编写的书籍或者文档。首先需要在根目录创建一个名为 `LANGS.md` 的文件，然后按照语言创建子目录：

```
# Languages  
* [中文](zh/)  
* [English](en/)  
* [French](fr/)  
* [Español](es/)
```

## 每种语言的配置

每个语言(例如：`en`)目录中都可以有一个 `book.json` 来定义自己的配置，它将作为主配置的扩展。

唯一的例外是插件，插件是全局指定的，语言环境配置不能指定特定的插件。

# Markdown语法

Markdown是GitBook的默认解析器，本文档基本上都是Markdown语法编写的。当然，你也可以选择[AsciiDoc语法](#)来编写文档。

下面是Markdown语法的概述。

## 标题

在文本之前添加一到六个 # 符号就可以创建一个标题。您使用的 # 号将决定标题的大小。

**这是1个#号的标题**

**这是2个#号的标题**

**这是3个#号的标题**

**这是4个#号的标题**

**这是5个#号的标题**

**这是6个#号的标题**

```
# 这是1个#号的标题
## 这是2个#号的标题
### 这是3个#号的标题
#### 这是4个#号的标题
##### 这是5个#号的标题
###### 这是6个#号的标题
```

GitBook支持一种显式方式设置头部ID。大括号中使用 # 号来设置ID值(大括号前必须有一个空格)，例如：

```
Hello {#id}
-----
# Hello {#id}
# Hello # {#id}
```

## 段落和换行符

段落是一个或多个连续的文本行，由一个或多个空白行分隔。

这里是我们开始的一条线。

此行与上面的一行通过两个换行符分隔，因此它将是一个  
\*单独的段落\*  
。

## 强调

此文本将为斜体这也将是*italic*

**此文本将是粗体这也将是bold**

~~这个文字会被划掉~~

You可以组合*them*

\*此文本将为斜体\*

这也将是italic

\*\*此文本将是粗体\*\*

这也将是bold

~~这个文字会被划掉~~

You \*\*可以\*\*组合*them*\_

## 列表

Markdown支持有序(编号)和无序(项目符号)列表。

### 无序

无序列表使用星号，加号和连字符(可互换)作为列表标记：

- 项目1
- 项目2
  - 项目2 a
  - 项目2 b

\*  
项目1

\*  
项目2

- \* 项目2a
- \* 项目2b

### 有序

有序列表使用数字后跟句点：

1. Item 1
2. Item 2
3. Item 3
  - Item 3a
  - Item 3b

1.  
Item 1

2.  
Item 2

3.  
Item 3

- \* Item 3a
- \* Item 3b

## 链接

Markdown支持两种类型的链接：内联和引用。

使用方括号包围文本并使用括号括住链接网址来创建简单的链接：

这是[带标题的链接（鼠标停留后显示标题）](#)与标题的内联链接。

[链接](#)没有标题属性。

```
这是[  
带标题的链接（鼠标停留后显示标题）  
](  
http://www.gibook.site/ "标题"  
)与标题的内联链接。  
  
[  
链接  
](  
http://www.gibook.site/  
)没有标题属性。
```

链接可以指向相对路径、页面定位或绝对网址。

### 其他方式

还有另一种方式来创建链接。标题使用引用名称定义，然后在方括号中使用此引用名称，而不是链接URL：

这是[一个示例](#)参考样式链接。

```
这是[  
一个示例  
][  
id  
]参考样式链接。
```

然后，在文档中的任何位置（一般放在文件结尾），定义您的链接标签：

```
[  
id  
]:http://www.gibook.site/ "可选标题这里"
```

这样，所有用 [链接标题][id] 的链接都会引用 [id]:http://www.gibook.site/ "可选标题这里" 这个地址，一般我们将。

## 图片

图像与链接创建的方式很类似：只需在方括号前使用感叹号即可：



图片:

```
图片:![  
这是图片  
](  
../images/logo_200.png  
)
```

## 引用块

使用 > 标记符后跟一个空格开始:

Kanye West说:

> 我们生活在未来 现在是我们的过去。

Kanye West说:

>  
我们生活在未来  
>  
现在是我们的过去。

## 表格

用连字符 - (第一行)分隔, 然后用管道符 | 分隔每个列来创建表格:

第一标题	第二标题
内容单元	内容单元
内容单元	内容单元

第一标题	第二标题
内容单元	内容单元
内容单元	内容单元

标题行的每一列中至少必须有三个连字符。

## 代码

Markdown支持两种不同的代码块样式。 第一种方式是缩进四个空格或一个 tab 的行, 而另一个种带小写波浪字符作为分隔符的行:

```
``` markdown  
这是一个示例代码块。
```

继续这里。

## 受防护的代码块

您可以通过在代码块之前和之后放置三个反引号`~~~~~`来创建围起来的代码块。我们建议在代码块之前和之后放置空行，以使原始格式化更容易阅读。

```
function test()
{
    console.log("注意这个函数之前的空行!")
}
```

```
```javascript
function test(){
    console.log("注意这个函数之前的空行? ")
}
```

```

## 语法高亮

您可以添加可选的语言标识符，以在受保护的代码块中启用语法突出显示。

例如，以语法高亮Ruby代码：

```
require  
'redcarpet'  
  
markdown = Redcarpet.new  
"Hello World!"  
)  
puts markdown.to_html
```

```
```ruby
require 'redcarpet'
markdown = Redcarpet.new("Hello World!")
puts markdown.to_html
```
```

内联代码

文本短语可以通过用反引号包围它们来标记为代码，下面的 `gitbook` 和 `text` 就是个例子：

使用`gitbook`在markdown中转换`text`语法到HTML。

脚注

GitBook支持简单的脚注语法。 脚注是对当前整个页面有效。

脚注参考前的文本。[a2](#)

```
脚注参考前的文本。[^2]
```

```
[  
^2  
]:评论要包括在脚注中。
```

## HTML

GitBook支持在您的文本中使用原始HTML，不处理HTML中的Markdown语法：

Markdown这里不会\*\*解析\*\*

```
<  
div  
>  
  
Markdown这里不会**解析**  
  
<  
/div  
>
```

## 分隔线

使用三个或多个星号、中划线、下划线创建分隔线：

---

---

---

三个或更多...

---

中划线

\*\*\*

星号

—

下划线

## 忽略Markdown格式

如果需要忽略Markdown格式，也就是转义Markdown的关键字，只需要在Markdown关键字前使用反斜杠 \ 即可。

```
Let's rename \*our-new-project\* to \*our-old-project\*.
```



# 生成电子书

GitBook不仅可以生成静态网站，也可以将内容输出为电子书(ePub, Mobi, PDF)格式。

```
# 生成PDF文件  
$ gitbook pdf ./ ./mybook.pdf  
  
# 生成ePub文件  
$ gitbook epub ./ ./mybook.epub  
  
# 生成Mobi文件  
$ gitbook mobi ./ ./mybook.mobi
```

## 安装ebook-convert

ebook-convert 是生成电子书所必需的(epub, mobi, pdf)插件。

### Linux系统

安装[Caliber应用程序](#)。

```
$ sudo aptitude install calibre
```

在某些Linux发行版中安装nodejs，您还需要手动创建一个nodejs软链接：

```
$ sudo ln -s /usr/bin/nodejs /usr/bin/node
```

### 苹果OS X系统

下载[Caliber应用程序](#)。将 `calibre.app` 移动到您的应用程序文件夹后，创建一个指向ebook-convert工具的软件链接：

```
$ sudo ln -s /Applications/calibre.app/Contents/MacOS/ebook-convert /usr/bin
```

这样就可以在任何目录下执行目录执行 `ebook-convert` 命令。

如果出现 `Operation not permitted` 异常，说明系统权限限制，需要配置环境变量的方式解决

```
$ sudo ln -s /Applications/calibre.app/Contents/MacOS/ebook-convert /usr/bin  
ln: /usr/bin/ebook-convert: Operation not permitted
```

### 环境变量配置

先启动ebook-convert完成第一次启动配置，然后关闭。接着在命令行窗口修改环境配置文件，加入 `EBOOK_PATH` (`ebook-convert`命令的所在目录)

```
vim ~/.bash_profile  
  
export  
EBOOK_PATH=/Applications/calibre.app/Contents/MacOS  
  
export  
PATH=  
$PATH  
:  
$EBOOK_PATH
```

然后刷新一下刚刚的配置：

```
source ~/.bash_profile
```

最后测试一下 `ebook-convert` 指令是否能正常被调用：

```
$ ebook-convert --version
ebook-convert (calibre 2.81.0)
Created by: Kovid Goyal
<kovid@kovidgoyal.net>
```

大功告成！下面就可以使用 `gitbook pdf ./ ./mybook.pdf` 命令把你的项目生成pdf文档了！

## 封面

所有格式电子书都可以创建封面。在安装完可以自己提供一个，或使用[autocover插件](#)生成一个。

要创建封面，请在您的书的根目录下放置一个 `cover.jpg` 文件。还可以添加 `cover_small.jpg` 指定一个较小版本的封面。  
封面图片格式必须是**JPEG**文件。

一个合格的封面应遵守以下准则：

- `cover.jpg` 的大小为1800x2360像素，`cover_small.jpg` 为200x262像素
- 无边框
- 清晰可见的书名
- 任何重要的文本应该在小版本中可见

## 生成PDF

进入文档项目目录，输入 `gitbook pdf ./ ./gitbook.pdf`

```
$ cd ~/gitbook-cn
$ gitbook pdf ./ ./gitbook.pdf
```

- `pdf`: 表示生成pdf格式，还有`epub`、`mobi`可选
- `./`: 表示需要生成书籍的项目根目录
- `./gitbook.pdf`: 表示生成书籍的名称
- 如果你的书籍有多种语言，就会生成多本书籍，书籍的名称会以语言结尾

# 成功案例

已经有超过50,000本图书使用[GitBook.com](#)发布。

## 文档

- DuckDuckHack 文档 by [DuckDuckGo](#)
- Loomio Handbook and guide to using Loomio both by [Loomio](#)
- Enspiral Handbook by [Enspiral](#)
- Webmagic开发文档 by <https://github.com/code4craft/webmagic>

## 图书

- Front-end Handbook by [Cody Lindley](#)
- How to make an Operating System by [@SamyPesse](#)
- Building Web Apps with Go by [@codegangsta](#)
- Django Girls Tutorial by [Django Girls](#)
- Linux Inside by [0xAx](#)
- Learn Javascript by [GitBook](#)

## 报告

- TowCenter Collection by [Columbia Journalism School](#)
- Block Relaxation Algorithms in Statistics by [Jan de Leeuw](#)

下面是GitBook的一些资源

- [GitBook主页](#)
- [Github地址](#)
- [GitBook编辑器](#)
- [GitBook Toolchain Documentation](#)
- [GitBook Documentation](#)