Hexo

# 什么是 Hexo？

Hexo 是一个快速、简洁且高效的博客框架。Hexo 使用 [Markdown](http://daringfireball.net/projects/markdown/)（或其他渲染引擎）解析文章，在几秒内，即可利用靓丽的主题生成静态网页。

# 安装

## 安装前提

需要先安装下列应用程序即可：

* [Node.js](http://nodejs.org/) (Node.js 版本需不低于 8.10，建议使用 Node.js 10.0 及以上版本)
* [Git](http://git-scm.com/)

下载node.js并安装（官网下载安装），安装过程和普通软一样，默认会安装npm。 控制台输入 node -v，控制台打印出对应Node版本就说明安装成功了

## 安装 Git

* Windows：下载并安装 [git](https://git-scm.com/download/win).
* Mac：使用 [Homebrew](http://mxcl.github.com/homebrew/), [MacPorts](http://www.macports.org/" \t "_blank) 或者下载 [安装程序](http://sourceforge.net/projects/git-osx-installer/)。
* Linux (Ubuntu, Debian)：sudo apt-get install git-core
* Linux (Fedora, Red Hat, CentOS)：sudo yum install git-core

**Mac 用户**

如果在编译时可能会遇到问题，请先到 App Store 安装 Xcode，Xcode 完成后，启动并进入 **Preferences -> Download -> Command Line Tools -> Install** 安装命令行工具。

**Windows 用户**

对于中国大陆地区用户，可以前往 [淘宝 Git for Windows 镜像](https://npm.taobao.org/mirrors/git-for-windows/) 下载 git 安装包。

## 安装 Node.js

Node.js 为大多数平台提供了官方的 [安装程序](https://nodejs.org/en/download/)。对于中国大陆地区用户，可以前往 [淘宝 Node.js 镜像](https://npm.taobao.org/mirrors/node) 下载。

其它的安装方法：

* Windows：通过 [nvs](https://github.com/jasongin/nvs/" \t "_blank)（推荐）或者[nvm](https://github.com/nvm-sh/nvm" \t "_blank) 安装。
* Mac：使用 [Homebrew](https://brew.sh/) 或 [MacPorts](http://www.macports.org/" \t "_blank) 安装。
* Linux（DEB/RPM-based）：从 [NodeSource](https://github.com/nodesource/distributions" \t "_blank) 安装。
* 其它：使用相应的软件包管理器进行安装，可以参考由 Node.js 提供的 [指导](https://nodejs.org/en/download/package-manager/)

对于 Mac 和 Linux 同样建议使用 nvs 或者 nvm，以避免可能会出现的权限问题。

**Windows 用户**

使用 Node.js 官方安装程序时，请确保勾选 **Add to PATH** 选项（默认已勾选）

**For Mac / Linux 用户**

如果在尝试安装 Hexo 的过程中出现 EACCES 权限错误，请遵循 [由 npmjs 发布的指导](https://docs.npmjs.com/resolving-eacces-permissions-errors-when-installing-packages-globally) 修复该问题。强烈建议 **不要** 使用 root、sudo 等方法覆盖权限

## 安装 Hexo

所有必备的应用程序安装完成后，即可使用 npm 安装 Hexo。

|  |
| --- |
| $ npm install -g hexo-cli |

## 进阶安装和使用

对于熟悉 npm 的进阶用户，可以仅局部安装 hexo 包。

|  |
| --- |
| $ npm install hexo |

安装以后，可以使用以下两种方式执行 Hexo：

1. npx hexo <command>
2. 将 Hexo 所在的目录下的 node\_modules 添加到环境变量之中即可直接使用 hexo <command>：

|  |
| --- |
| echo 'PATH="$PATH:./node\_modules/.bin"' >> ~/.profile |

# 建站

安装 Hexo 完成后，请执行下列命令，Hexo 将会在指定文件夹中新建所需要的文件。

|  |
| --- |
| $ hexo init <folder> $ cd <folder> $ npm install |

新建完成后，指定文件夹的目录如下：

.  
├── \_config.yml   
├── package.json  
├── scaffolds  
├── source  
| ├── \_drafts  
| └── \_posts  
└── themes

|  |
| --- |
|  |

## Hexo文件夹介绍

node\_modules 依赖包

scaffolds 命令生成文章等的模板

source 用命令创建的各种文章

themes 主题

\_config.yml 整个博客的配置

package.json 项目所需模块项目的配置信息

## config.yml

网站的 [配置](https://hexo.io/zh-cn/docs/configuration) 信息，您可以在此配置大部分的参数。

## package.json

应用程序的信息。[EJS](https://ejs.co/" \t "_blank), [Stylus](http://learnboost.github.io/stylus/) 和 [Markdown](http://daringfireball.net/projects/markdown/) renderer 已默认安装，您可以自由移除。

package.json

|  |
| --- |
| {  "name": "hexo-site",  "version": "0.0.0",  "private": true,  "hexo": {  "version": ""  },  "dependencies": {  "hexo": "^3.8.0",  "hexo-generator-archive": "^0.1.5",  "hexo-generator-category": "^0.1.3",  "hexo-generator-index": "^0.2.1",  "hexo-generator-tag": "^0.2.0",  "hexo-renderer-ejs": "^0.3.1",  "hexo-renderer-stylus": "^0.3.3",  "hexo-renderer-marked": "^0.3.2",  "hexo-server": "^0.3.3"  } } |

## scaffolds

[模版](https://hexo.io/zh-cn/docs/writing) 文件夹。当您新建文章时，Hexo 会根据 scaffold 来建立文件。

Hexo的模板是指在新建的文章文件中默认填充的内容。例如，如果您修改scaffold/post.md中的Front-matter内容，那么每次新建一篇文章时都会包含这个修改。

## source

资源文件夹是存放用户资源的地方。除 \_posts 文件夹之外，开头命名为 \_ (下划线)的文件 / 文件夹和隐藏的文件将会被忽略。Markdown 和 HTML 文件会被解析并放到 public 文件夹，而其他文件会被拷贝过去。

## themes

[主题](https://hexo.io/zh-cn/docs/themes) 文件夹。Hexo 会根据主题来生成静态页面。

1. npm install hexo-cli -g
2. hexo init blog
3. cd blog
4. npm install
5. hexo server

打开浏览器，输入[http://localhost:4000](https://link.juejin.im/?target=http%3A%2F%2Flocalhost%3A4000) ，看到以下页面说明Hexo环境安装成功。

## [Hexo常用命令](https://link.juejin.im/?target=https%3A%2F%2Fhexo.io%2Fzh-cn%2Fdocs%2Fcommands.html)

npm install hexo -g *#安装*

npm update hexo -g *#升级*

hexo init *#初始化*

hexo n "我的博客" == hexo new "我的博客" *#新建文章*

hexo p == hexo publish

hexo g == hexo generate*#生成*

hexo s == hexo server *#启动服务预览*

2. hexo d == hexo deploy*#部署*
4. hexo server *#Hexo 会监视文件变动并自动更新，您无须重启服务器。*
6. hexo server -s *#静态模式*
8. hexo server -p 5000 *#更改端口*
10. hexo server -i 192.168.1.1 *#自定义 IP*
12. hexo clean *#清除缓存 网页正常情况下可以忽略此条命令*
14. hexo g *#生成静态网页*
16. hexo d *#开始部署*

# 配置

可以在 \_config.yml 中修改大部分的配置。

## 网站

| **参数** | **描述** |
| --- | --- |
| title | 网站标题 |
| subtitle | 网站副标题 |
| description | 网站描述 |
| keywords | 网站的关键词。使用半角逗号 , 分隔多个关键词。 |
| author | 您的名字 |
| language | 网站使用的语言。对于简体中文用户来说，使用不同的主题可能需要设置成不同的值，请参考你的主题的文档自行设置，常见的有 zh-Hans和 zh-CN。 |
| timezone | 网站时区。Hexo 默认使用您电脑的时区。请参考 [时区列表](https://en.wikipedia.org/wiki/List_of_tz_database_time_zones) 进行设置，如 America/New\_York, Japan, 和 UTC 。一般的，对于中国大陆地区可以使用 Asia/Shanghai。 |

其中，description主要用于SEO，告诉搜索引擎一个关于您站点的简单描述，通常建议在其中包含您网站的关键词。author参数用于主题显示文章的作者。

## 网址

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| url | 网址 |  |
| root | 网站根目录 |  |
| permalink | 文章的 [永久链接](https://hexo.io/zh-cn/docs/permalinks) 格式 | :year/:month/:day/:title/ |
| permalink\_defaults | 永久链接中各部分的默认值 |  |
| pretty\_urls | 改写 [permalink](https://hexo.io/zh-cn/docs/variables) 的值来美化 URL |  |
| pretty\_urls.trailing\_index | 是否在永久链接中保留尾部的 index.html，设置为 false 时去除 | true |
| pretty\_urls.trailing\_html | 是否在永久链接中保留尾部的 .html, 设置为 false 时去除 (对尾部的 *index.html*无效) | true |

**网站存放在子目录**

如果您的网站存放在子目录中，例如 http://yoursite.com/blog，则请将您的 url 设为 http://yoursite.com/blog 并把 root 设为 /blog/。

例如：

|  |
| --- |
| # 比如，一个页面的永久链接是 http://example.com/foo/bar/index.html pretty\_urls:  trailing\_index: false # 此时页面的永久链接会变为 http://example.com/foo/bar/ |

## 目录

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| source\_dir | 资源文件夹，这个文件夹用来存放内容。 | source |
| public\_dir | 公共文件夹，这个文件夹用于存放生成的站点文件。 | public |
| tag\_dir | 标签文件夹 | tags |
| archive\_dir | 归档文件夹 | archives |
| category\_dir | 分类文件夹 | categories |
| code\_dir | Include code 文件夹，source\_dir 下的子目录 | downloads/code |
| i18n\_dir | 国际化（i18n）文件夹 | :lang |
| skip\_render | 跳过指定文件的渲染。匹配到的文件将会被不做改动地复制到 public 目录中。您可使用 [glob 表达式](https://github.com/micromatch/micromatch#extended-globbing)来匹配路径。 |  |

例如：

|  |
| --- |
| skip\_render: "mypage/\*\*/\*" # 将会直接将 `source/mypage/index.html` 和 `source/mypage/code.js` 不做改动地输出到 'public' 目录 # 你也可以用这种方法来跳过对指定文章文件的渲染 skip\_render: "\_posts/test-post.md" # 这将会忽略对 'test-post.md' 的渲染 |

**提示**

如果您刚刚开始接触 Hexo，通常没有必要修改这一部分的值。

## 文章

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| new\_post\_name | 新文章的文件名称 | :title.md |
| default\_layout | 预设布局 | post |
| auto\_spacing | 在中文和英文之间加入空格 | false |
| titlecase | 把标题转换为 title case | false |
| external\_link | 在新标签中打开链接 | true |
| external\_link.enable | 在新标签中打开链接 | true |
| external\_link.field | 对整个网站（site）生效或仅对文章（post）生效 | site |
| external\_link.exclude | 需要排除的域名。主域名和子域名如 www 需分别配置 | [] |
| filename\_case | 把文件名称转换为 (1) 小写或 (2) 大写 | 0 |
| render\_drafts | 显示草稿 | false |
| post\_asset\_folder | 启动 [Asset 文件夹](https://hexo.io/zh-cn/docs/asset-folders) | false |
| relative\_link | 把链接改为与根目录的相对位址 | false |
| future | 显示未来的文章 | true |
| highlight | 代码块的设置 |  |
| highlight.enable | 开启代码块高亮 | true |
| highlight.auto\_detect | 如果未指定语言，则启用自动检测 | false |
| highlight.line\_number | 显示行数 Enabling this option will also enable *wrap* option | true |
| highlight.tab\_replace | 用 n 个空格替换 tabs；如果值为空，则不会替换 tabs | '' |
| highlight.wrap | Wrap the code block in [<table>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table) | true |
| highlight.hljs | Use the hljs-\* prefix for CSS classes | false |

**相对地址**

默认情况下，Hexo 生成的超链接都是绝对地址。例如，如果您的网站域名为 example.com,您有一篇文章名为 hello，那么绝对链接可能像这样：http://example.com/hello.html，它是**绝对**于域名的。相对链接像这样：/hello.html，也就是说，无论用什么域名访问该站点，都没有关系，这在进行反向代理时可能用到。通常情况下，建议使用绝对地址。

## 分类 & 标签

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| default\_category | 默认分类 | uncategorized |
| category\_map | 分类别名 |  |
| tag\_map | 标签别名 |  |

## 日期 / 时间格式

Hexo 使用 [Moment.js](http://momentjs.com/) 来解析和显示时间。

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| date\_format | 日期格式 | YYYY-MM-DD |
| time\_format | 时间格式 | HH:mm:ss |
| use\_date\_for\_updated | 启用以后，如果 Front Matter 中没有指定 updated， [post.updated](https://hexo.io/zh-cn/docs/variables" \l "%E9%A1%B5%E9%9D%A2%E5%8F%98%E9%87%8F) 将会使用 date 的值而不是文件的创建时间。在 Git 工作流中这个选项会很有用 | true |

## 分页

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| per\_page | 每页显示的文章量 (0 = 关闭分页功能) | 10 |
| pagination\_dir | 分页目录 | page |

## 扩展

| **参数** | **描述** |
| --- | --- |
| theme | 当前主题名称。值为false时禁用主题 |
| theme\_config | 主题的配置文件。在这里放置的配置会覆盖主题目录下的 \_config.yml 中的配置 |
| deploy | 部署部分的设置 |
| meta\_generator | [Meta generator](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/meta#%E5%B1%9E%E6%80%A7) 标签。 值为 false 时 Hexo 不会在头部插入该标签 |

### 包括或不包括目录和文件

在 Hexo 配置文件中，通过设置 include/exclude 可以让 Hexo 进行处理或忽略某些目录和文件夹。你可以使用 [glob 表达式](https://github.com/isaacs/minimatch) 对目录和文件进行匹配。

include and exclude options only apply to the source/ folder, whereas ignore option applies to all folders.

| **参数** | **描述** |
| --- | --- |
| include | Hexo 默认会忽略隐藏文件和文件夹（包括名称以下划线和 . 开头的文件和文件夹，Hexo 的 \_posts 和 \_data 等目录除外）。通过设置此字段将使 Hexo 处理他们并将它们复制到 source 目录下。 |
| exclude | Hexo 会忽略这些文件和目录 |
| ignore | Ignore files/folders |

举例：

|  |
| --- |
| # Include/Exclude Files/Folders include:  - ".nojekyll"  # 包括 'source/css/\_typing.css'  - "css/\_typing.css"  # 包括 'source/\_css/' 中的任何文件，但不包括子目录及其其中的文件。  - "\_css/\*"  # 包含 'source/\_css/' 中的任何文件和子目录下的任何文件  - "\_css/\*\*/\*"  exclude:  # 不包括 'source/js/test.js'  - "js/test.js"  # 不包括 'source/js/' 中的文件、但包括子目录下的所有目录和文件  - "js/\*"  # 不包括 'source/js/' 中的文件和子目录下的任何文件  - "js/\*\*/\*"  # 不包括 'source/js/' 目录下的所有文件名以 'test' 开头的文件，但包括其它文件和子目录下的单文件  - "js/test\*"  # 不包括 'source/js/' 及其子目录中任何以 'test' 开头的文件  - "js/\*\*/test\*"  # 不要用 exclude 来忽略 'source/\_posts/' 中的文件。你应该使用 'skip\_render'，或者在要忽略的文件的文件名之前加一个下划线 '\_'  # 在这里配置一个 - "\_posts/hello-world.md" 是没有用的。  ignore:  # Ignore any folder named 'foo'.  - "\*\*/foo"  # Ignore 'foo' folder in 'themes/' only.  - "\*\*/themes/\*/foo"  # Same as above, but applies to every subfolders of 'themes/'.  - "\*\*/themes/\*\*/foo" |

列表中的每一项都必须用单引号或双引号包裹起来。

include 和 exclude 并不适用于 themes/ 目录下的文件。如果需要忽略 themes/ 目录下的部分文件或文件夹，可以使用 ignore 或在文件名之前添加下划线 \_。

### 使用代替配置文件

可以在 hexo-cli 中使用 --config 参数来指定自定义配置文件的路径。你可以使用一个 YAML 或 JSON 文件的路径，也可以使用逗号分隔（无空格）的多个 YAML 或 JSON 文件的路径。例如：

|  |
| --- |
| # use 'custom.yml' in place of '\_config.yml' $ hexo server --config custom.yml  # use 'custom.yml' & 'custom2.json', prioritizing 'custom3.yml', then 'custom2.json' $ hexo generate --config custom.yml,custom2.json,custom3.yml |

当你指定了多个配置文件以后，Hexo 会按顺序将这部分配置文件合并成一个 \_multiconfig.yml。如果遇到重复的配置，排在后面的文件的配置会覆盖排在前面的文件的配置。这个原则适用于任意数量、任意深度的 YAML 和 JSON 文件。

例如，使用 --options 指定了两个自定义配置文件：

|  |
| --- |
| $ hexo generate --config custom.yml,custom2.json |

如果 custom.yml 中指定了 foo: bar，在 custom2.json 中指定了 "foo": "dinosaur"，那么在 \_multiconfig.yml 中你会得到 foo: dinosaur。

### 覆盖主题配置

通常情况下，Hexo 主题是一个独立的项目，并拥有一个独立的 \_config.yml 配置文件。  
你可以在站点的 \_config.yml 配置文件中配置你的主题，这样你就不需要 fork 一份主题并维护主题独立的配置文件。

以下是一个覆盖主题配置的例子：

|  |
| --- |
| # \_config.yml theme\_config:  bio: "My awesome bio" |

|  |
| --- |
| # themes/my-theme/\_config.yml bio: "Some generic bio" logo: "a-cool-image.png" |

最终主题配置的输出是：

|  |
| --- |
| {  bio: "My awesome bio",  logo: "a-cool-image.png" } |

# hexo部署到github

1、[安装git](https://links.jianshu.com/go?to=https%3A%2F%2Fgit-scm.com%2Fdownload%2Fwin" \t "_blank)

2、[安装nodejs](https://links.jianshu.com/go?to=https%3A%2F%2Fnodejs.org%2Fen%2F" \t "_blank)

3、用 npm 安装话经常出现卡住而导致无法正常安装，解决办法就是修改 npm 的安装源，这里选择淘宝 NPM 镜像，这是一个完整 npmjs.org 镜像，你可以用此代替官方版本，同步频率目前为 10分钟 一次以保证尽量与官方服务同步。

npm config set registry https://registry.npm.taobao.org

4、安装hexo

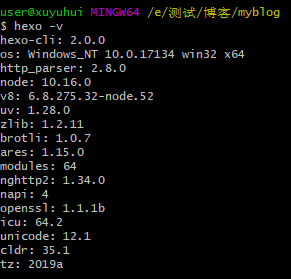
npm install -g hexo-cli



安装hexo

5、查看版本信息

hexo -v



查看版本信息

6、初始化hexo程序

hexo init myblog

（更新npm/并更换为国内的npm镜像命令：npm install npm -g/  
npm install -g cnpm --registry=[https://registry.npm.taobao.org](https://links.jianshu.com/go?to=https%3A%2F%2Fregistry.npm.taobao.org)）

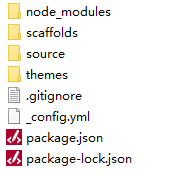
7、进入myblog文件夹

cd myblog

（8、npm install）

9、查看文件夹目录

* node\_modules: 依赖包
* scaffolds：生成文章的一些模板
* source：用来存放你的文章
* themes：主题
* \*\* \_config.yml: 博客的配置文件\*\*



文件夹目录

10、打开hexo

hexo g

hexo server

11、创建以用于提交

git config --global user.name "yourname"

git config --global user.email "youremail"

12、连接公钥

ssh-keygen -t rsa -C "email"

13、查看是否成功

ssh -T git@github.com

（You've successfully authenticated, but GitHub does not provide shell access.）

14、将hexo和GitHub关联起来，修改配置文件\_config.yml

deploy:

type: （中间有个空格，以下相同）git

repo: https://github.com/YourgithubName/YourgithubName.github.io.git

branch: master

15、安装deploy-git部署命令

npm install hexo-deployer-git --save

16、清除之前生成的东西

hexo clean

17、生成静态文章

hexo generate/hexo g

18、部署文章

hexo deploy/hexo d

19、github中要新建一个仓库用户名+github.io，访问用户名.github.io即可

## 设置hexo主题

1、[选择hexo主题](https://links.jianshu.com/go?to=https%3A%2F%2Fhexo.io%2Fthemes%2F" \t "_blank)，到github上下载主题放到themes文件夹下  
2、修改\_config.yml里的theme为该文件夹名称

## 新增菜单栏选项

1、添加新页面：hexo new page "xx"  
2、在主题配置文件的menu中加上该页面  
3、在zh-CN.yml文件中加上中文意思

## 新增草稿

1、添加草稿：hexo new draft "xx"  
2、预览草稿：hexo server --draft  
3、发布草稿：hexo publish draft "xx"

## 图片处理

1、配置文件中的post\_asset\_folder改成true  
2、npm install hexo-asset-image --save  
3、放到该文件夹下，按照md格式输入(文件名称/图片名.jpg)

### 搜索

1、npm install hexo-generator-searchdb --save  
2、站点配置文件的扩展下添加

search:

path: search.xml

field: post

format: html

limit: 10000

3、主题配置文件下，local\_search改成true即可

local\_search:

enable: true

### 字数

1、npm install hexo-symbols-count-time --save  
2、站点配置

symbols\_count\_time:

symbols: true

time: true

total\_symbols: true

total\_time: true

### 宠物

1、npm install --save hexo-helper-live2d  
2、[live2d插件](https://links.jianshu.com/go?to=https%3A%2F%2Fhuaji8.top%2Fpost%2Flive2d-plugin-2.0%2F" \t "_blank)  
3、安装合适的宠物npm install live2d-widget-model-wanko  
4、在配置文件中添加即可

live2d:

enable: true

scriptFrom: local

pluginRootPath: live2dw/

pluginJsPath: lib/

pluginModelPath: assets/

model:

# use: live2d-widget-model-tororo

# use: live2d-widget-model-hijiki

use: live2d-widget-model-wanko

display:

position: right

width: 150

height: 300

# 水平位置

# hOffset: 0

# 垂直位置

# vOffset: -20

mobile:

show: true

### 点击弹爱心

1、在source->js目录下新建一个js文件为click-love.js

! function (e, t, a) {

function n() {

c(".heart{width: 10px;height: 10px;position: fixed;background: #f00;transform: rotate(45deg);-webkit-transform: rotate(45deg);-moz-transform: rotate(45deg);}.heart:after,.heart:before{content: '';width: inherit;height: inherit;background: inherit;border-radius: 50%;-webkit-border-radius: 50%;-moz-border-radius: 50%;position: fixed;}.heart:after{top: -5px;}.heart:before{left: -5px;}"), o(), r()

}

function r() {

for (var e = 0; e < d.length; e++) d[e].alpha <= 0 ? (t.body.removeChild(d[e].el), d.splice(e, 1)) : (d[e].y--, d[e].scale += .004, d[e].alpha -= .013, d[e].el.style.cssText = "left:" + d[e].x + "px;top:" + d[e].y + "px;opacity:" + d[e].alpha + ";transform:scale(" + d[e].scale + "," + d[e].scale + ") rotate(45deg);background:" + d[e].color + ";z-index:99999");

requestAnimationFrame(r)

}

function o() {

var t = "function" == typeof e.onclick && e.onclick;

e.onclick = function (e) {

t && t(), i(e)

}

}

function i(e) {

var a = t.createElement("div");

a.className = "heart", d.push({

el: a,

x: e.clientX - 5,

y: e.clientY - 5,

scale: 1,

alpha: 1,

color: s()

}), t.body.appendChild(a)

}

function c(e) {

var a = t.createElement("style");

a.type = "text/css";

try {

a.appendChild(t.createTextNode(e))

} catch (t) {

a.styleSheet.cssText = e

}

t.getElementsByTagName("head")[0].appendChild(a)

}

function s() {

return "rgb(" + ~~(255 \* Math.random()) + "," + ~~(255 \* Math.random()) + "," + ~~(255 \* Math.random()) + ")"

}

var d = [];

e.requestAnimationFrame = function () {

return e.requestAnimationFrame || e.webkitRequestAnimationFrame || e.mozRequestAnimationFrame || e.oRequestAnimationFrame || e.msRequestAnimationFrame || function (e) {

setTimeout(e, 1e3 / 60)

}

}(), n()

}(window, document);

2、在主题theme->layout->\_partials->footer文件中导入该js文件即可

<script type="text/javascript" src="/js/click-love.js"></script>

# 内容

你可以执行下列命令来创建一篇新文章或者新的页面。

|  |
| --- |
| $ hexo new [layout] <title> |

您可以在命令中指定文章的布局（layout），默认为 post，可以通过修改 \_config.yml 中的 default\_layout 参数来指定默认布局。

## 布局（Layout）

Hexo 有三种默认布局：post、page 和 draft。在创建者三种不同类型的文件时，它们将会被保存到不同的路径；而您自定义的其他布局和 post 相同，都将储存到 source/\_posts 文件夹。

| **布局** | **路径** |
| --- | --- |
| post | source/\_posts |
| page | source |
| draft | source/\_drafts |

**不要处理我的文章**

如果你不想你的文章被处理，你可以将 Front-Matter 中的layout: 设为 false 。

## 文件名称

Hexo 默认以标题做为文件名称，但您可编辑 new\_post\_name 参数来改变默认的文件名称，举例来说，设为 :year-:month-:day-:title.md 可让您更方便的通过日期来管理文章。

| **变量** | **描述** |
| --- | --- |
| :title | 标题（小写，空格将会被替换为短杠） |
| :year | 建立的年份，比如， 2015 |
| :month | 建立的月份（有前导零），比如， 04 |
| :i\_month | 建立的月份（无前导零），比如， 4 |
| :day | 建立的日期（有前导零），比如， 07 |
| :i\_day | 建立的日期（无前导零），比如， 7 |

## 草稿

刚刚提到了 Hexo 的一种特殊布局：draft，这种布局在建立时会被保存到 source/\_drafts 文件夹，您可通过 publish 命令将草稿移动到 source/\_posts 文件夹，该命令的使用方式与 new 十分类似，您也可在命令中指定 layout 来指定布局。

|  |
| --- |
| $ hexo publish [layout] <title> |

草稿默认不会显示在页面中，您可在执行时加上 --draft 参数，或是把 render\_drafts 参数设为 true 来预览草稿。

# 服务器

## [hexo-server](https://github.com/hexojs/hexo-server)

Hexo 3.0 把服务器独立成了个别模块，您必须先安装 [hexo-server](https://github.com/hexojs/hexo-server" \t "_blank) 才能使用。

|  |
| --- |
| $ npm install hexo-server --save |

安装完成后，输入以下命令以启动服务器，您的网站会在 http://localhost:4000 下启动。在服务器启动期间，Hexo 会监视文件变动并自动更新，您无须重启服务器。

|  |
| --- |
| $ hexo server |

如果您想要更改端口，或是在执行时遇到了 EADDRINUSE 错误，可以在执行时使用 -p 选项指定其他端口，如下：

|  |
| --- |
| $ hexo server -p 5000 |

### 静态模式

在静态模式下，服务器只处理 public 文件夹内的文件，而不会处理文件变动，在执行时，您应该先自行执行 hexo generate，此模式通常用于生产环境（production mode）下。

|  |
| --- |
| $ hexo server -s |

### 自定义 IP

服务器默认运行在 0.0.0.0，您可以覆盖默认的 IP 设置，如下：

|  |
| --- |
| $ hexo server -i 192.168.1.1 |

指定这个参数后，您就只能通过该IP才能访问站点。例如，对于一台使用无线网络的笔记本电脑，除了指向本机的127.0.0.1外，通常还有一个192.168.\*.\*的局域网IP，如果像上面那样使用-i参数，就不能用127.0.0.1来访问站点了。对于有公网IP的主机，如果您指定一个局域网IP作为-i参数的值，那么就无法通过公网来访问站点。

## Pow

[Pow](http://pow.cx/) 是一个 Mac 系统上的零配置 Rack 服务器，它也可以作为一个简单易用的静态文件服务器来使用。

### 安装

|  |
| --- |
| $ curl get.pow.cx | sh |

### 设置

在 ~/.pow 文件夹建立链接（symlink）。

|  |
| --- |
| $ cd ~/.pow $ ln -s /path/to/myapp |

您的网站将会在 http://myapp.dev 下运行，网址根据链接名称而定。

## 生成文件

使用 Hexo 生成静态文件快速而且简单。

|  |
| --- |
| $ hexo generate |

#### 监视文件变动

Hexo 能够监视文件变动并立即重新生成静态文件，在生成时会比对文件的 SHA1 checksum，只有变动的文件才会写入。

|  |
| --- |
| $ hexo generate --watch |

#### 完成后部署

您可执行下列的其中一个命令，让 Hexo 在生成完毕后自动部署网站，两个命令的作用是相同的。

|  |
| --- |
| $ hexo generate --deploy $ hexo deploy --generate |

**简写**

上面两个命令可以简写为  
$ hexo g -d  
$ hexo d -g

## 将 Hexo 部署到 GitHub Pages

在本教程中，我们将会使用 [Travis CI](https://travis-ci.com/) 将 Hexo 博客部署到 GitHub Pages 上。Travis CI 对于开源 repository 是免费的，但是这意味着你的站点文件将会是公开的。如果你希望你的站点文件不被公开，请直接前往本文 [私有 Repository](#私有 Repository) 部分。

1. 新建一个 repository。如果你希望你的站点能通过 <你的 GitHub 用户名>.github.io 域名访问，你的 repository 应该直接命名为 <你的 GitHub 用户名>.github.io。
2. 将你的 Hexo 站点文件夹推送到 repository 中。默认情况下不应该 public 目录将不会被推送到 repository 中，你应该检查 .gitignore 文件中是否包含 public 一行，如果没有请加上。
3. 将 [Travis CI](https://github.com/marketplace/travis-ci) 添加到你的 GitHub 账户中。
4. 前往 GitHub 的 [Applications settings](https://github.com/settings/installations)，配置 Travis CI 权限，使其能够访问你的 repository。
5. 你应该会被重定向到 Travis CI 的页面。如果没有，请 [手动前往](https://travis-ci.com/)。
6. 在浏览器新建一个标签页，前往 GitHub [新建 Personal Access Token](https://github.com/settings/tokens)，只勾选 repo 的权限并生成一个新的 Token。Token 生成后请复制并保存好。
7. 回到 Travis CI，前往你的 repository 的设置页面，在 **Environment Variables** 下新建一个环境变量，**Name** 为 GH\_TOKEN，**Value** 为刚才你在 GitHub 生成的 Token。确保 **DISPLAY VALUE IN BUILD LOG**保持 **不被勾选** 避免你的 Token 泄漏。点击 **Add** 保存。
8. 在你的 Hexo 站点文件夹中新建一个 .travis.yml 文件：

|  |
| --- |
| sudo: false language: node\_js node\_js:  - 10 # use nodejs v10 LTS cache: npm branches:  only:  - master # build master branch only script:  - hexo generate # generate static files deploy:  provider: pages  skip-cleanup: true  github-token: $GH\_TOKEN  keep-history: true  on:  branch: master  local-dir: public |

1. 将 .travis.yml 推送到 repository 中。Travis CI 应该会自动开始运行，并将生成的文件推送到同一 repository 下的 gh-pages 分支下
2. 在 GitHub 中前往你的 repository 的设置页面，修改 GitHub Pages 的部署分支为 gh-pages。
3. 前往 https://<你的 GitHub 用户名>.github.io 查看你的站点是否可以访问。这可能需要一些时间。

### Project page

如果你更希望你的站点部署在 <你的 GitHub 用户名>.github.io 的子目录中，你的 repository 需要直接命名为子目录的名字，这样你的站点可以通过 https://<你的 GitHub 用户名>.github.io/<repository 的名字> 访问。你需要检查你的 Hexo 配置文件，将 url 修改为 https://<你的 GitHub 用户名>.github.io/<repository 的名字>、将 root 的值修改为 /<repository 的名字>/

### 私有 Repository

下面的指示基于 [一键部署](https://hexo.io/docs/one-command-deployment) 编写。

1. 安装 [hexo-deployer-git](https://github.com/hexojs/hexo-deployer-git" \t "_blank).
2. 在 **\_config.yml**（如果有已存在的请删除）添加如下配置：

|  |
| --- |
| deploy:  type: git  repo: https://github.com/<username>/<project>  # example, https://github.com/hexojs/hexojs.github.io  branch: gh-pages |

1. 运行 hexo clean && hexo deploy 。
2. 查看 username.github.io 上的网页是否部署成功。

### 有用的参考链接

* [GitHub Pages 使用文档](https://help.github.com/categories/github-pages-basics/)
* [Travis CI 使用文档](https://docs.travis-ci.com/user/tutorial/)
* [Awesome Hexo](https://github.com/hexojs/awesome-hexo)
* [在百度上搜索 “Hexo GitHub”](https://www.baidu.com/s?wd=Hexo%20GitHub)

## 将 Hexo 部署到 GitLab Pages

在本教程中，我们将会使用 GitLab CI 将 Hexo 博客部署到 GitLab Pages 上。

1. 新建一个 repository。如果你希望你的站点能通过 <你的 GitLab 用户名>.gitlab.io 域名访问，你的 repository 应该直接命名为 <你的 GitLab 用户名>.gitlab.io。
2. 将你的 Hexo 站点文件夹推送到 repository 中。默认情况下不应该 public 目录将不会被推送到 repository 中，你应该检查 .gitignore 文件中是否包含 public 一行，如果没有请加上。
3. 在你的站点文件夹中新建 .gitlab-ci.yml 文件：

|  |
| --- |
| image: node:10-alpine # use nodejs v10 LTS cache:  paths:  - node\_modules/  before\_script:  - npm install hexo-cli -g  - npm install  pages:  script:  - hexo generate  artifacts:  paths:  - public  only:  - master |

1. GitLab CI 应该会自动开始运行，构建成功以后你应该可以在 https://<你的 GitLab 用户名>.gitlab.io 查看你的网站。
2. (可选) 如果你需要查看生成的文件，可以在 [job artifact](https://docs.gitlab.com/ee/user/project/pipelines/job_artifacts.html) 中找到。

在 GitLab.com 上，GitLab CI 是默认启用的。如果你使用的是自托管的 GitLab，你可能需要在 Settings -> CI / CD -> Shared Runners 启用 GitLab CI。

### Project page

如果你更希望你的站点部署在 <你的 GitLab 用户名>.gitlab.io 的子目录中，你的 repository 需要直接命名为子目录的名字，这样你的站点可以通过 https://<你的 GitLab 用户名>.gitlab.io/<repository 的名字> 访问。你需要检查你的 Hexo 配置文件，将 url 的值修改为 https://<你的 GitLab 用户名>.gitlab.io/<repository 的名字>、将 root 的值修改为 /<repository 的名字>/

### Useful links

* [GitLab Pages 相关文档](https://docs.gitlab.com/ee/user/project/pages/index.html)
* [GitLab CI 相关文档](https://docs.gitlab.com/ee/ci/README.html)
* [在百度上搜索 “Hexo GitLab”](https://www.baidu.com/s?wd=Hexo%20GitLab)

## 部署

Hexo 提供了快速方便的一键部署功能，让您只需一条命令就能将网站部署到服务器上。

|  |
| --- |
| $ hexo deploy |

在开始之前，您必须先在 \_config.yml 中修改参数，一个正确的部署配置中至少要有 type 参数，例如：

|  |
| --- |
| deploy:  type: git |

您可同时使用多个 deployer，Hexo 会依照顺序执行每个 deployer。

|  |
| --- |
| deploy: - type: git  repo: - type: heroku  repo: |

Refer to the [Plugins](https://hexo.io/plugins/) list for more deployment plugins.

**缩进**

YAML依靠缩进来确定元素间的从属关系。因此，请确保每个deployer的缩进长度相同，并且使用空格缩进。

### Git

1. 安装 [hexo-deployer-git](https://github.com/hexojs/hexo-deployer-git" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-git --save |

1. 修改配置。

|  |
| --- |
| deploy:  type: git  repo: <repository url> #https://bitbucket.org/JohnSmith/johnsmith.bitbucket.io  branch: [branch]  message: [message] |

| **参数** | **描述** | **默认** |
| --- | --- | --- |
| repo | 库（Repository）地址 |  |
| branch | 分支名称 | gh-pages (GitHub) coding-pages (Coding.net) master (others) |
| message | 自定义提交信息 | Site updated: {{ now('YYYY-MM-DD HH:mm:ss') }}) |
| token | Optional token value to authenticate with the repo. Prefix with $ to read token from environment variable |  |

1. 生成站点文件并推送至远程库。执行 hexo clean && hexo deploy。
   * You will be prompted with username and password of the target repository, unless you authenticate with a token or ssh key.
   * hexo-deployer-git does not store your username and password. Use [git-credential-cache](https://git-scm.com/docs/git-credential-cache) to store them temporarily.
2. 登入 Github/BitBucket/Gitlab，请在库设置（Repository Settings）中将默认分支设置为\_config.yml配置中的分支名称。稍等片刻，您的站点就会显示在您的Github Pages中。

#### 这一切是如何发生的？

当执行 hexo deploy 时，Hexo 会将 public 目录中的文件和目录推送至 \_config.yml 中指定的远端仓库和分支中，并且**完全覆盖**该分支下的已有内容。

**For 使用 Git 管理站点目录的用户**

由于 Hexo 的部署默认使用分支 master，所以如果你同时正在使用 Git 管理你的站点目录，你应当注意你的部署分支应当不同于写作分支。  
一个好的实践是将站点目录和 Pages 分别存放在两个不同的 Git 仓库中，可以有效避免相互覆盖。  
Hexo 在部署你的站点生成的文件时并不会更新你的站点目录。因此你应该手动提交并推送你的写作分支。

此外，如果您的 Github Pages 需要使用 CNAME 文件**自定义域名**，请将 CNAME 文件置于 source 目录下，只有这样 hexo deploy 才能将 CNAME 文件一并推送至部署分支。

### Heroku

安装 [hexo-deployer-heroku](https://github.com/hexojs/hexo-deployer-heroku" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-heroku --save |

修改配置。

|  |
| --- |
| deploy:  type: heroku  repo: <repository url>  message: [message] |

| **参数** | **描述** |
| --- | --- |
| repo | Heroku 库（Repository）地址 |
| message | 自定提交信息 (默认为 Site updated: {{ now('YYYY-MM-DD HH:mm:ss') }}) |

### Netlify

[Netlify](https://www.netlify.com/) 是一个提供网络托管的综合平台。它集持续集成（CI）CDN 自定义域名 HTTPS 持续部署（CD）等诸多功能于一身。您可以通过以下两种方式将Hexo站点部署到Netlify。

首先，也是最通用的方式，就是使用Netlify提供的网页端用户界面。前往[新建一个网站页面](https://app.netlify.com/start" \t "_blank)，选择需要关联的 Github/BitBucket/Gitlab 库，然后遵循网站提示。

另一种方式是使用Netlify提供的命令行客户端工具 [Node based CLI](https://www.netlify.com/docs/cli/) 管理和部署您的站点。

此外，您还可以在项目的README中增加一个 [部署至Netlify按钮](https://www.netlify.com/docs/deploy-button/)，这样其他用户在fork或clone了您的项目之后可以方便快捷地一键部署。

### Rsync

安装 [hexo-deployer-rsync](https://github.com/hexojs/hexo-deployer-rsync" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-rsync --save |

修改配置。

|  |
| --- |
| deploy:  type: rsync  host: <host>  user: <user>  root: <root>  port: [port]  delete: [true|false]  verbose: [true|false]  ignore\_errors: [true|false] |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| host | 远程主机的地址 |  |
| user | 使用者名称 |  |
| root | 远程主机的根目录 |  |
| port | 端口 | 22 |
| delete | 删除远程主机上的旧文件 | true |
| verbose | 显示调试信息 | true |
| ignore\_errors | 忽略错误 | false |

**rsync部署模块的工作方式**

需要注意的是，要求您提供的实际上是一个能通过SSH登陆远程主机的Linux用户。Hexo会自动处理关于rsync使用的一切操作。因此，您需要在远程主机上为您的Hexo站点建立一个用户，并允许其通过SSH登陆。不过，这里的port，的确是指rsync监听的端口，请确保防火墙打开了该端口。

### OpenShift

安装 [hexo-deployer-openshift](https://github.com/hexojs/hexo-deployer-openshift" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-openshift --save |

修改配置。

|  |
| --- |
| deploy:  type: openshift  repo: <repository url>  message: [message] |

| **参数** | **描述** |
| --- | --- |
| repo | OpenShift 库（Repository）地址 |
| message | 自定提交信息 (默认为 Site updated: {{ now('YYYY-MM-DD HH:mm:ss') }}) |

### FTPSync

安装 [hexo-deployer-ftpsync](https://github.com/hexojs/hexo-deployer-ftpsync" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-ftpsync --save |

修改配置。

|  |
| --- |
| deploy:  type: ftpsync  host: <host>  user: <user>  pass: <password>  remote: [remote]  port: [port]  ignore: [ignore]  connections: [connections]  verbose: [true|false] |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| host | 远程主机的地址 |  |
| user | 使用者名称 |  |
| pass | 密码 |  |
| remote | 远程主机的根目录 | / |
| port | 端口 | 21 |
| ignore | 忽略的文件或目录 |  |
| connections | 使用的连接数 | 1 |
| verbose | 显示调试信息 | false |

**FTP部署可能出现的问题**

您可能需要预先通过其他方式将所有文件上传到远程主机中。否则初次使用ftpsync插件就可能出现报错。另外，由于FTP协议的特征，它每传送一个文件就需要一次握手，相对速度较慢。

### SFTP

安装 [hexo-deployer-sftp](https://github.com/lucascaro/hexo-deployer-sftp" \t "_blank)。

|  |
| --- |
| $ npm install hexo-deployer-sftp --save |

修改配置。

|  |
| --- |
| deploy:  type: sftp  host: <host>  user: <user>  pass: <password>  remotePath: [remote path]  port: [port]  privateKey: [path/to/privateKey]  passphrase: [passphrase]  agent: [path/to/agent/socket] |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| host | 远程主机的地址 |  |
| user | 使用者名称 |  |
| pass | 密码 |  |
| remotePath | 远程主机的根目录 | / |
| port | 端口 | 22 |
| privateKey | ssh私钥的目录地址 |  |
| passphrase | （可省略）ssh私钥的密码短语 |  |
| agent | ssh套接字的目录地址 | $SSH\_AUTH\_SOCK |

### ZEIT Now

[ZEIT Now](https://zeit.co/) 是一个托管静态网站和 Serverless APIs 的云平台。你可以一键将你的 Hexo 网站部署到 ZEIT Now、并使用自己的域名或者由 ZEIT Now 提供的免费子域名 now.sh。

1. 安装 [Now CLI](https://zeit.co/download):

|  |
| --- |
| $ npm i -g now |

1. 在 package.json 中添加 npm script:

|  |
| --- |
| {  "scripts": {  "build": "hexo generate"  } } |

1. 在 Hexo 所在目录执行下述指令：

|  |
| --- |
| now |

你也可以使用下面的按钮在 ZEIT Now 创建一个 Hexo 项目。

### 21云盒子

1. 在 [21云盒子](https://www.21yunbox.com/) 中， 创建一个新的 静态网页，然后使用以下设置：

* **构建命令:** yarn && hexo deploy
* **发布目录:** public

1. 点击 “部署” 按钮！

样例已经部署在 <https://hexo.21yunbox.com/>.

请查看 [用 21云盒子 部署一个 Hexo 样例](https://www.21yunbox.com/docs/#/deploy-hexo).

### 其他方法

Hexo 生成的所有文件都放在 public 文件夹中，您可以将它们复制到您喜欢的地方。

## 永久链接（Permalinks）

可以在 \_config.yml 配置中调整网站的永久链接或者在每篇文章的 Front-matter 中指定。

#### 变量

除了下列变量外，您还可使用 Front-matter 中的所有属性。

| **变量** | **描述** |
| --- | --- |
| :year | 文章的发表年份（4 位数） |
| :month | 文章的发表月份（2 位数） |
| :i\_month | 文章的发表月份（去掉开头的零） |
| :day | 文章的发表日期 (2 位数) |
| :i\_day | 文章的发表日期（去掉开头的零） |
| :hour | 文章发表时的小时 (2 位数) |
| :minute | 文章发表时的分钟 (2 位数) |
| :title | 文件名称 |
| :post\_title | 文章标题 |
| :id | 文章 ID (not persistent across [*cache reset*](https://hexo.io/zh-cn/docs/commands#clean)) |
| :category | 分类。如果文章没有分类，则是 default\_category 配置信息。 |

您可在 permalink\_defaults 参数下调整永久链接中各变量的默认值：

|  |
| --- |
| permalink\_defaults:  lang: en |

#### 示例

假设 source/\_posts 文件夹中有个 hello-world.md，包含以下内容：

|  |
| --- |
| title: Hello World date: 2013-07-14 17:01:34 categories: - foo - bar |

| **参数** | **结果** |
| --- | --- |
| :year/:month/:day/:title/ | 2013/07/14/hello-world/ |
| :year-:month-:day-:title.html | 2013-07-14-hello-world.html |
| :category/:title/ | foo/bar/hello-world/ |

#### 多语种支持

若要建立一个多语种的网站，您可修改 new\_post\_name 和 permalink 参数，如下：

new\_post\_name: :lang/:title.md

permalink: :lang/:title/

当您建立新文章时，文章会被储存到：

|  |
| --- |
| $ hexo new "Hello World" --lang tw  # => source/\_posts/tw/Hello-World.md |

而网址会是：

|  |
| --- |
| http://localhost:4000/tw/hello-world/ |

# Hexo迁移

## RSS

首先，安装 hexo-migrator-rss 插件。

|  |
| --- |
| $ npm install hexo-migrator-rss --save |

插件安装完成后，执行下列命令，从 RSS 迁移所有文章。source 可以是文件路径或网址。

|  |
| --- |
| $ hexo migrate rss <source> |

## Jekyll

把 \_posts 文件夹内的所有文件复制到 source/\_posts 文件夹，并在 \_config.yml 中修改 new\_post\_name 参数。

|  |
| --- |
| new\_post\_name: :year-:month-:day-:title.md |

## Octopress

把 Octopress source/\_posts 文件夹内的所有文件转移到 Hexo 的 source/\_posts 文件夹，并修改 \_config.yml 中的 new\_post\_name 参数。

|  |
| --- |
| new\_post\_name: :year-:month-:day-:title.md |

## WordPress

首先，安装 hexo-migrator-wordpress 插件。

|  |
| --- |
| $ npm install hexo-migrator-wordpress --save |

在 WordPress 仪表盘中导出数据(“Tools” → “Export” → “WordPress”)（详情参考[WP支持页面](http://en.support.wordpress.com/export/" \t "_blank)）。

插件安装完成后，执行下列命令来迁移所有文章。source 可以是 WordPress 导出的文件路径或网址。

|  |
| --- |
| $ hexo migrate wordpress <source> |

**注意**

这个插件并不能完美地实现WordPress->Hexo的数据转换，尤其是在处理WordPress的分类方面存在问题（见[Front-matter中的分类与标签](https://hexo.io/zh-cn/docs/front-matter)）。因此，建议您在迁移完成后，手工审阅所有生成的markdown文件，检查其中是否有错误。对于文章数量较大的WordPress站点，这项工作可能要花很长的时间。

## Joomla

首先，安装 hexo-migrator-joomla 插件。

|  |
| --- |
| $ npm install hexo-migrator-joomla --save |

使用 [J2XML](http://extensions.joomla.org/extensions/migration-a-conversion/data-import-a-export/12816?qh=YToxOntpOjA7czo1OiJqMnhtbCI7fQ%253D%253D) 组件导出 Joomla 文章。  
插件安装完成后，执行下列命令来迁移所有文章。source 可以是 Joomla 导出的文件路径或网址。

|  |
| --- |
| $ hexo migrate joomla <source> |

# 主题

创建 Hexo 主题非常容易，您只要在 themes 文件夹内，新增一个任意名称的文件夹，并修改 \_config.yml 内的 theme 设定，即可切换主题。一个主题可能会有以下的结构：

|  |
| --- |
| . ├── \_config.yml ├── languages ├── layout ├── scripts └── source |

## \_config.yml

主题的配置文件。和 Hexo 配置文件不同，主题配置文件修改时会自动更新，无需重启 Hexo Server。

## languages

语言文件夹。请参见 [国际化 (i18n)](https://hexo.io/zh-cn/docs/internationalization)。

## layout

布局文件夹。用于存放主题的模板文件，决定了网站内容的呈现方式，Hexo 内建 [Swig](https://github.com/node-swig/swig-templates) 模板引擎，您可以另外安装插件来获得 [EJS](https://github.com/hexojs/hexo-renderer-ejs)、[Haml](https://github.com/hexojs/hexo-renderer-haml" \t "_blank)、[Jade](https://github.com/hexojs/hexo-renderer-jade" \t "_blank) 或 [Pug](https://github.com/maxknee/hexo-render-pug) 支持，Hexo 根据模板文件的扩展名来决定所使用的模板引擎，例如：

|  |
| --- |
| layout.ejs - 使用 EJS layout.swig - 使用 Swig |

您可参考 [模板](https://hexo.io/zh-cn/docs/templates) 以获得更多信息。

## scripts

脚本文件夹。在启动时，Hexo 会载入此文件夹内的 JavaScript 文件，请参见 [插件](https://hexo.io/zh-cn/docs/plugins) 以获得更多信息。

## source

资源文件夹，除了模板以外的 Asset，例如 CSS、JavaScript 文件等，都应该放在这个文件夹中。文件或文件夹开头名称为 \_（下划线线）或隐藏的文件会被忽略。

如果文件可以被渲染的话，会经过解析然后储存到 public 文件夹，否则会直接拷贝到 public 文件夹。

## 发布

当您完成主题后，可以考虑将它发布到 [主题列表](https://hexo.io/themes)，让更多人能够使用您的主题。在发布前建议先进行 [主题单元测试](https://github.com/hexojs/hexo-theme-unit-test)，确保每一项功能都能正常使用。发布主题的步骤和 [更新文档](https://hexo.io/zh-cn/docs/contributing#%E6%9B%B4%E6%96%B0%E6%96%87%E6%A1%A3) 非常类似。

1. Fork [hexojs/site](https://github.com/hexojs/site" \t "_blank)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

|  |
| --- |
| $ git clone https://github.com/<username>/site.git $ cd site $ npm install |

1. 编辑 source/\_data/themes.yml，在文件中新增您的主题，例如：

|  |
| --- |
| - name: landscape  description: A brand new default theme for Hexo.  link: https://github.com/hexojs/hexo-theme-landscape  preview: http://hexo.io/hexo-theme-landscape  tags:  - official  - responsive  - widget  - two\_column  - one\_column |

1. 在 source/themes/screenshots 新增同名的截图档案，图片必须为 800x500 的 PNG 文件。
2. 推送（push）分支。
3. 建立一个新的合并申请（pull request）并描述改动。

## 模版（Scaffold）

在新建文章时，Hexo 会根据 scaffolds 文件夹内相对应的文件来建立文件，例如：

|  |
| --- |
| $ hexo new photo "My Gallery" |

在执行这行指令时，Hexo 会尝试在 scaffolds 文件夹中寻找 photo.md，并根据其内容建立文章，以下是您可以在模版中使用的变量：

| **变量** | **描述** |
| --- | --- |
| layout | 布局 |
| title | 标题 |
| date | 文件建立日期 |

模板决定了网站内容的呈现方式，每个主题至少都应包含一个 index 模板，以下是各页面相对应的模板名称：

| **模板** | **用途** | **回退** |
| --- | --- | --- |
| index | 首页 |  |
| post | 文章 | index |
| page | 分页 | index |
| archive | 归档 | index |
| category | 分类归档 | archive |
| tag | 标签归档 | archive |
|  |  |  |

### 支持的格式

Hexo 支持以任何格式书写文章，只要安装了相应的渲染插件。

例如，Hexo 默认安装了 hexo-renderer-marked 和 hexo-renderer-ejs，因此你不仅可以用 Markdown 写作，你还可以用 EJS 写作。如果你安装了 hexo-renderer-pug，你甚至可以用 Pug 模板语言书写文章。

只需要将文章的扩展名从 md 改成 ejs，Hexo 就会使用 hexo-renderer-ejs 渲染这个文件，其他格式同理。

abbrlink

### Front-matter

Front-matter 是文件最上方以 --- 分隔的区域，用于指定个别文件的变量，举例来说：

|  |
| --- |
| --- title: Hello World date: 2013/7/13 20:46:25 --- |

以下是预先定义的参数，您可在模板中使用这些参数值并加以利用。

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| layout | 布局 |  |
| title | 标题 | 文章的文件名 |
| date | 建立日期 | 文件建立日期 |
| updated | 更新日期 | 文件更新日期 |
| comments | 开启文章的评论功能 | true |
| tags | 标签（不适用于分页） |  |
| categories | 分类（不适用于分页） |  |
| permalink | 覆盖文章网址 |  |
| keywords | 仅用于 meta 标签和 Open Graph 的关键词（不推荐使用） |  |

#### 分类和标签

只有文章支持分类和标签，您可以在 Front-matter 中设置。在其他系统中，分类和标签听起来很接近，但是在 Hexo 中两者有着明显的差别：分类具有顺序性和层次性，也就是说 Foo, Bar 不等于 Bar, Foo；而标签没有顺序和层次。

|  |
| --- |
| categories: - Diary tags: - PS3 - Games |

**分类方法的分歧**

如果您有过使用 WordPress 的经验，就很容易误解 Hexo 的分类方式。WordPress 支持对一篇文章设置多个分类，而且这些分类可以是同级的，也可以是父子分类。但是 Hexo 不支持指定多个同级分类。下面的指定方法：

|  |
| --- |
| categories:  - Diary  - Life |

会使分类Life成为Diary的子分类，而不是并列分类。因此，有必要为您的文章选择尽可能准确的分类。

如果你需要为文章添加多个分类，可以尝试以下 list 中的方法。

|  |
| --- |
| categories: - [Diary, PlayStation] - [Diary, Games] - [Life] |

此时这篇文章同时包括三个分类： PlayStation 和 Games 分别都是父分类 Diary 的子分类，同时 Life 是一个没有子分类的分类。

#### JSON Front-matter

除了 YAML 外，你也可以使用 JSON 来编写 Front-matter，只要将 --- 代换成 ;;; 即可。

|  |
| --- |
| "title": "Hello World", "date": "2013/7/13 20:46:25" ;;; |

### 布局（Layout）

如果页面结构类似，例如两个模板都有页首（Header）和页脚（Footer），您可考虑通过「布局」让两个模板共享相同的结构。一个布局文件必须要能显示 body 变量的内容，如此一来模板的内容才会被显示，举例来说：

index.ejs

|  |
| --- |
| index |

layout.ejs

|  |
| --- |
| <!DOCTYPE html> <html>  <body><%- body %></body> </html> |

生成：

|  |
| --- |
| <!DOCTYPE html> <html>  <body>index</body> </html> |

每个模板都默认使用 layout 布局，您可在 front-matter 指定其他布局，或是设为 false 来关闭布局功能，您甚至可在布局中再使用其他布局来建立嵌套布局。

### 局部模版（Partial）

局部模板让您在不同模板之间共享相同的组件，例如页首（Header）、页脚（Footer）或侧边栏（Sidebar）等，可利用局部模板功能分割为个别文件，让维护更加便利。举例来说：

partial/header.ejs

|  |
| --- |
| <h1 id="logo"><%= config.title %></h1> |

index.ejs

|  |
| --- |
| <%- partial('partial/header') %> <div id="content">Home page</div> |

生成：

|  |
| --- |
| <h1 id="logo">My Site</h1> <div id="content">Home page</div> |

#### 局部变量

您可以在局部模板中指定局部变量并使用。

partial/header.ejs

|  |
| --- |
| <h1 id="logo"><%= title %></h1> |

index.ejs

|  |
| --- |
| <%- partial('partial/header', {title: 'Hello World'}) %> <div id="content">Home page</div> |

生成：

|  |
| --- |
| <h1 id="logo">Hello World</h1> <div id="content">Home page</div> |

### 标签插件（Tag Plugins）

标签插件和 Front-matter 中的标签不同，它们是用于在文章中快速插入特定内容的插件。

虽然你可以使用任何格式书写你的文章，但是标签插件永远可用，且语法也都是一致的。

#### 引用块

在文章中插入引言，可包含作者、来源和标题。

**别号：** quote

|  |
| --- |
| {% blockquote [author[, source]] [link] [source\_link\_title] %} content {% endblockquote %} |

##### 样例

**没有提供参数，则只输出普通的 blockquote**

|  |
| --- |
| {% blockquote %} Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque hendrerit lacus ut purus iaculis feugiat. Sed nec tempor elit, quis aliquam neque. Curabitur sed diam eget dolor fermentum semper at eu lorem. {% endblockquote %} |

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque hendrerit lacus ut purus iaculis feugiat. Sed nec tempor elit, quis aliquam neque. Curabitur sed diam eget dolor fermentum semper at eu lorem.

**引用书上的句子**

|  |
| --- |
| {% blockquote David Levithan, Wide Awake %} Do not just seek happiness for yourself. Seek happiness for all. Through kindness. Through mercy. {% endblockquote %} |

Do not just seek happiness for yourself. Seek happiness for all. Through kindness. Through mercy.

**David Levithan**Wide Awake

**引用 Twitter**

|  |
| --- |
| {% blockquote @DevDocs https://twitter.com/devdocs/status/356095192085962752 %} NEW: DevDocs now comes with syntax highlighting. http://devdocs.io {% endblockquote %} |

NEW: DevDocs now comes with syntax highlighting. [http://devdocs.io](http://devdocs.io/)

**@DevDocs**[*twitter.com/devdocs/status/356095192085962752*](https://twitter.com/devdocs/status/356095192085962752)

**引用网络上的文章**

|  |
| --- |
| {% blockquote Seth Godin http://sethgodin.typepad.com/seths\_blog/2009/07/welcome-to-island-marketing.html Welcome to Island Marketing %} Every interaction is both precious and an opportunity to delight. {% endblockquote %} |

Every interaction is both precious and an opportunity to delight.

**Seth Godin**[*Welcome to Island Marketing*](http://sethgodin.typepad.com/seths_blog/2009/07/welcome-to-island-marketing.html)

#### 代码块

在文章中插入代码。

**别名：** code

|  |
| --- |
| {% codeblock [title] [lang:language] [url] [link text] [additional options] %} code snippet {% endcodeblock %} |

Specify additional options in option:value format, e.g. line\_number:false first\_line:5.

| **Extra Options** | **Description** | **Default** |
| --- | --- | --- |
| line\_number | Show line number | true |
| highlight | Enable code highlighting | true |
| first\_line | Specify the first line number | 1 |
| mark | Line highlight specific line(s), each value separated by a comma. Specify number range using a dash Example: mark:1,4-7,10 will mark line 1, 4 to 7 and 10. |  |
| wrap | Wrap the code block in [<table>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/table) | true |

##### 样例

**普通的代码块**

|  |
| --- |
| {% codeblock %} alert('Hello World!'); {% endcodeblock %} |

|  |
| --- |
| alert('Hello World!'); |

**指定语言**

|  |
| --- |
| {% codeblock lang:objc %} [rectangle setX: 10 y: 10 width: 20 height: 20]; {% endcodeblock %} |

|  |
| --- |
| [rectangle setX: 10 y: 10 width: 20 height: 20]; |

**附加说明**

|  |
| --- |
| {% codeblock Array.map %} array.map(callback[, thisArg]) {% endcodeblock %} |

Array.map

|  |
| --- |
| array.map(callback[, thisArg]) |

**附加说明和网址**

|  |
| --- |
| {% codeblock \_.compact http://underscorejs.org/#compact Underscore.js %} \_.compact([0, 1, false, 2, '', 3]); => [1, 2, 3] {% endcodeblock %} |

\_.compact[Underscore.js](http://underscorejs.org/" \l "compact" \t "_blank)

|  |
| --- |
| \_.compact([0, 1, false, 2, '', 3]); => [1, 2, 3] |

#### 反引号代码块

另一种形式的代码块，不同的是它使用三个反引号来包裹。

``` [language] [title] [url] [link text] code snippet ```

#### Pull Quote

在文章中插入 Pull quote。

|  |
| --- |
| {% pullquote [class] %} content {% endpullquote %} |

#### jsFiddle

在文章中嵌入 jsFiddle。

|  |
| --- |
| {% jsfiddle shorttag [tabs] [skin] [width] [height] %} |

#### Gist

在文章中嵌入 Gist。

|  |
| --- |
| {% gist gist\_id [filename] %} |

#### iframe

在文章中插入 iframe。

|  |
| --- |
| {% iframe url [width] [height] %} |

#### Image

在文章中插入指定大小的图片。

|  |
| --- |
| {% img [class names] /path/to/image [width] [height] '"title text" "alt text"' %} |

#### Link

在文章中插入链接，并自动给外部链接添加 target="\_blank" 属性。

|  |
| --- |
| {% link text url [external] [title] %} |

#### Include Code

插入 source/downloads/code 文件夹内的代码文件。source/downloads/code 不是固定的，取决于你在配置文件中 code\_dir 的配置。

|  |
| --- |
| {% include\_code [title] [lang:language] [from:line] [to:line] path/to/file %} |

##### 样例

**嵌入 test.js 文件全文**

|  |
| --- |
| {% include\_code lang:javascript test.js %} |

**只嵌入第 3 行**

|  |
| --- |
| {% include\_code lang:javascript from:3 to:3 test.js %} |

**嵌入第 5 行至第 8 行**

|  |
| --- |
| {% include\_code lang:javascript from:5 to:8 test.js %} |

**嵌入第 5 行至文件结束**

|  |
| --- |
| {% include\_code lang:javascript from:5 test.js %} |

**嵌入第 1 行至第 8 行**

|  |
| --- |
| {% include\_code lang:javascript to:8 test.js %} |

#### Youtube

在文章中插入 Youtube 视频。

|  |
| --- |
| {% youtube video\_id %} |

#### Vimeo

在文章中插入 Vimeo 视频。

|  |
| --- |
| {% vimeo video\_id %} |

#### 引用文章

引用其他文章的链接。

|  |
| --- |
| {% post\_path filename %} {% post\_link filename [title] [escape] %} |

在使用此标签时可以忽略文章文件所在的路径或者文章的永久链接信息、如语言、日期。

例如，在文章中使用 {% post\_link how-to-bake-a-cake %} 时，只需有一个名为 how-to-bake-a-cake.md 的文章文件即可。即使这个文件位于站点文件夹的 source/posts/2015-02-my-family-holiday 目录下、或者文章的永久链接是 2018/en/how-to-bake-a-cake，都没有影响。

默认链接文字是文章的标题，你也可以自定义要显示的文本。此时不应该使用 Markdown 语法 []()。

默认对文章的标题和自定义标题里的特殊字符进行转义。可以使用escape选项，禁止对特殊字符进行转义。

**链接使用文章的标题**

{% post\_link hexo-3-8-released %}

[Hexo 3.8.0 Released](https://hexo.io/news/2018/10/19/hexo-3-8-released/)

**链接使用自定义文字**

{% post\_link hexo-3-8-released '通往文章的链接' %}

[通往文章的链接](https://hexo.io/news/2018/10/19/hexo-3-8-released/)

**对标题的特殊字符进行转义**

|  |
| --- |
| {% post\_link hexo-4-released 'How to use <b> tag in title' %} |

[How to use <b> tag in title](https://hexo.io/news/2019/10/14/hexo-4-released/)

**禁止对标题的特殊字符进行转义**

|  |
| --- |
| {% post\_link hexo-4-released '<b>bold</b> custom title' false %} |

[bold custom title](https://hexo.io/news/2019/10/14/hexo-4-released/)

#### 引用资源

引用文章的资源。

|  |
| --- |
| {% asset\_path filename %} {% asset\_img filename [title] %} {% asset\_link filename [title] [escape] %} |

#### Raw

如果您想在文章中插入 Swig 标签，可以尝试使用 Raw 标签，以免发生解析异常。

|  |
| --- |
| {% raw %} content {% endraw %} |

#### 文章摘要和截断

在文章中使用 <!-- more -->，那么 <!-- more --> 之前的文字将会被视为摘要。首页中将只出现这部分文字，同时这部分文字也会出现在正文之中。

例如：

|  |
| --- |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. <!-- more --> Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. |

首页中将只会出现

|  |
| --- |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. |

正文中则会出现

|  |
| --- |
| Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. |

注意，摘要可能会被 Front Matter 中的 excerpt 覆盖。

### 资源文件夹

资源（Asset）代表 source 文件夹中除了文章以外的所有文件，例如图片、CSS、JS 文件等。比方说，如果你的Hexo项目中只有少量图片，那最简单的方法就是将它们放在 source/images 文件夹中。然后通过类似于 ![](/images/image.jpg) 的方法访问它们。

#### 文章资源文件夹

对于那些想要更有规律地提供图片和其他资源以及想要将他们的资源分布在各个文章上的人来说，Hexo也提供了更组织化的方式来管理资源。这个稍微有些复杂但是管理资源非常方便的功能可以通过将 config.yml文件中的 post\_asset\_folder 选项设为 true 来打开。

\_config.yml

|  |
| --- |
| post\_asset\_folder: true |

当资源文件管理功能打开后，Hexo将会在你每一次通过 hexo new [layout] <title> 命令创建新文章时自动创建一个文件夹。这个资源文件夹将会有与这个文章文件一样的名字。将所有与你的文章有关的资源放在这个关联文件夹中之后，你可以通过相对路径来引用它们，这样你就得到了一个更简单而且方便得多的工作流。

#### 相对路径引用的标签插件

通过常规的 markdown 语法和相对路径来引用图片和其它资源可能会导致它们在存档页或者主页上显示不正确。在Hexo 2时代，社区创建了很多插件来解决这个问题。但是，随着Hexo 3 的发布，许多新的标签插件被加入到了核心代码中。这使得你可以更简单地在文章中引用你的资源。

|  |
| --- |
| {% asset\_path slug %} {% asset\_img slug [title] %} {% asset\_link slug [title] %} |

比如说：当你打开文章资源文件夹功能后，你把一个 example.jpg 图片放在了你的资源文件夹中，如果通过使用相对路径的常规 markdown 语法 ![](/example.jpg) ，它将 不会 出现在首页上。（但是它会在文章中按你期待的方式工作）

正确的引用图片方式是使用下列的标签插件而不是 markdown ：

|  |
| --- |
| {% asset\_img example.jpg This is an example image %} |

通过这种方式，图片将会同时出现在文章和主页以及归档页中。

### 数据文件

有时您可能需要在主题中使用某些资料，而这些资料并不在文章内，并且是需要重复使用的，那么您可以考虑使用 Hexo 3.0 新增的「数据文件」功能。此功能会载入 source/\_data 内的 YAML 或 JSON 文件，如此一来您便能在网站中复用这些文件了。

举例来说，在 source/\_data 文件夹中新建 menu.yml 文件：

|  |
| --- |
| Home: / Gallery: /gallery/ Archives: /archives/ |

您就能在模板中使用这些资料：

|  |
| --- |
| <% for (var link in site.data.menu) { %>  <a href="<%= site.data.menu[link] %>"> <%= link %> </a> <% } %> |

渲染结果如下 :

|  |
| --- |
| <a href="/"> Home </a> <a href="/gallery/"> Gallery </a> <a href="/archives/"> Archives </a> |

### 优化

如果您的主题太过于复杂，或是需要生成的文件量太过于庞大，可能会大幅降低性能，除了简化主题外，您可以考虑 Hexo 2.7 新增的局部缓存（Fragment Caching） 功能。

本功能借鉴于 [Ruby on Rails](http://guides.rubyonrails.org/caching_with_rails.html#fragment-caching)，它储存局部内容，下次便能直接使用缓存内容，可以减少文件夹查询并使生成速度更快。

它可用于页首、页脚、侧边栏等文件不常变动的位置，举例来说：

|  |
| --- |
| <%- fragment\_cache('header', function(){  return '<header></header>'; }); |

如果您使用局部模板的话，可以更简单：

|  |
| --- |
| <%- partial('header', {}, {cache: true}); |

fragment\_cache() 将会缓存第一次的渲染结果，并在之后直接输出缓存的结果。因此只有在不同页面的渲染结果都相同时才应使用局部缓存。  
比如，在配置中启用了 relative\_link 后不应该使用局部缓存，因为相对链接在每个页面可能不同。

## Hexo变量

#### 全局变量

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| site | [网站变量](https://hexo.io/zh-cn/docs/variables#%E7%BD%91%E7%AB%99%E5%8F%98%E9%87%8F) | object; 见 [网站变量](https://hexo.io/zh-cn/docs/variables#%E7%BD%91%E7%AB%99%E5%8F%98%E9%87%8F) |
| page | 针对该页面的内容以及 front-matter 中自定义的变量。 | object; 见 [页面变量](https://hexo.io/zh-cn/docs/variables#%E9%A1%B5%E9%9D%A2%E5%8F%98%E9%87%8F) |
| config | 网站配置 | object (站点的配置文件) |
| theme | 主题配置。继承自网站配置。 | object (主题配置文件) |
| \_ (单下划线) | [Lodash](https://lodash.com/) 函数库 | [Lodash](https://lodash.com/) 文档 |
| path | 当前页面的路径（不含根路径） | string |
| url | 当前页面的完整网址 | string |
| env | 环境变量 | ??? |

#### 网站变量

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| site.posts | 所有文章 | array of post objects |
| site.pages | 所有分页 | array of page objects |
| site.categories | 所有分类 | object，包含了站点全部的分类 |
| site.tags | 所有标签 | array，包含了站点全部的标签 |

#### 页面变量

**页面（page）**

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.title | 页面标题 | string |
| page.date | 页面建立日期 | [Moment.js](http://momentjs.com/) 对象 |
| page.updated | 页面更新日期 | [Moment.js](http://momentjs.com/) 对象 |
| page.comments | 留言是否开启 | boolean |
| page.layout | 布局名称 | string |
| page.content | 页面的完整内容 | string |
| page.excerpt | 页面摘要 | string |
| page.more | 除了页面摘要的其余内容 | string |
| page.source | 页面原始路径 | string |
| page.full\_source | 页面的完整原始路径 | string |
| page.path | 页面网址（不含根路径）。我们通常在主题中使用 url\_for(page.path)。 | string |
| page.permalink | 页面的完整网址 | string |
| page.prev | 上一个页面。如果此为第一个页面则为 null。 | string or null |
| page.next | 下一个页面。如果此为最后一个页面则为 null。 | string or null |
| page.raw | 文章的原始内容 | ??? |
| page.photos | 文章的照片（用于相簿） | array |
| page.link | 文章的外部链接（用于链接文章） | string |

**文章 (post):** 与 page 布局相同，但新增以下变量。

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.published | 如果该文章已发布则为 true | boolean |
| page.categories | 该文章的所有分类 | array of ??? |
| page.tags | 该文章的所有标签 | array of ??? |

**首页（index）**

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.per\_page | 每页显示的文章数量 | number |
| page.total | 总页数 | number |
| page.current | 目前页数 | number |
| page.current\_url | 目前分页的网址 | string |
| page.posts | 本页文章 ([Data Model](https://hexojs.github.io/warehouse/)) | object |
| page.prev | 上一页的页数。如果此页是第一页的话则为 0。 | number |
| page.prev\_link | 上一页的网址。如果此页是第一页的话则为 ''。 | string |
| page.next | 下一页的页数。如果此页是最后一页的话则为 0。 | number |
| page.next\_link | 下一页的网址。如果此页是最后一页的话则为 ''。 | string |
| page.path | 当前页面的路径（不含根目录）。我们通常在主题中使用 url\_for(page.path)。 | string |

**归档 (archive)**：与 index 布局相同，但新增以下变量。

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.archive | 等于 true | boolean |
| page.year | 年份归档 (4位) | number |
| page.month | 月份归档 (没有前导零的2位数) | number |

**分类 (category)**：与 index 布局相同，但新增以下变量。

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.category | 分类名称 | string |

**标签 (tag)**：与 index 布局相同，但新增以下变量。

| **变量** | **描述** | **类型** |
| --- | --- | --- |
| page.tag | 标签名称 | string |

## 辅助函数（Helpers）

辅助函数帮助您在模版中快速插入内容。辅助函数不能在源文件中使用。

### 网址

#### url\_for

在路径前加上根路径，从 Hexo 2.7 开始您应该使用此函数而不是 config.root + path。

|  |
| --- |
| <%- url\_for(path, [option]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| relative | 是否输出相对链接 | config.relative\_link 的值 |

**示例：**

|  |
| --- |
| \_config.yml root: /blog/ |

|  |
| --- |
| <%- url\_for('/a/path') %> // /blog/a/path |

是否输出相对链接，默认遵循配置文件中 relative\_link 的值  
例如， post/page 的相对路径值可能是 /foo/bar/index.html

|  |
| --- |
| \_config.yml relative\_link: true |

|  |
| --- |
| <%- url\_for('/css/style.css') %> // ../../css/style.css /\* 覆盖配置  \* 即使配置文件中启用了 relative\_link，你也可以使用 relative 参数禁用相对链接输出，反之亦然  \*/ <%- url\_for('/css/style.css', {relative: false}) %> // /css/style.css |

#### relative\_url

取得与 from 相对的 to 路径。

|  |
| --- |
| <%- relative\_url(from, to) %> |

**示例：**

|  |
| --- |
| <%- relative\_url('foo/bar/', 'css/style.css') %> // ../../css/style.css |

#### gravatar

根据邮箱地址返回 Gravatar 头像 URL。

如果你不指定 options 参数，将会应用默认参数。否则，你可以将其设置为一个数字，这个数字将会作为 Gravatar 的大小参数。最后，如果你设置它一个对象，它将会被转换为 Gravatar 的一个查询字符串参数。

|  |
| --- |
| <%- gravatar(email, [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| s | 图片大小 | 80 |
| d | 默认头像 |  |
| f | 强制使用默认图象 |  |
| r | 头像等级限制 |  |

访问 [Gravatar](https://en.gravatar.com/site/implement/images/) 了解更多。

**示例：**

|  |
| --- |
| <%- gravatar('a@abc.com') %> // https://www.gravatar.com/avatar/b9b00e66c6b8a70f88c73cb6bdb06787  <%- gravatar('a@abc.com', 40) %> // https://www.gravatar.com/avatar/b9b00e66c6b8a70f88c73cb6bdb06787?s=40  <%- gravatar('a@abc.com' {s: 40, d: 'https://via.placeholder.com/150'}) %> // https://www.gravatar.com/avatar/b9b00e66c6b8a70f88c73cb6bdb06787?s=40&d=https%3A%2F%2Fvia.placeholder.com%2F150 |

#### full\_url\_for

在路径前加上根路径和域名。输出会被自动转码。

|  |
| --- |
| <%- full\_url\_for(path) %> |

**示例：**

|  |
| --- |
| \_config.yml url: https://example.com/blog # example |

|  |
| --- |
| <%- full\_url\_for('/a/path') %> // https://example.com/blog/a/path |

### HTML 标签

#### css

载入 CSS 文件。path 可以是数组或字符串，如果 path 开头不是 / 或任何协议，则会自动加上根路径；如果后面没有加上 .css 扩展名的话，也会自动加上。Use object type for custom attributes.

|  |
| --- |
| <%- css(path, ...) %> |

**示例：**

|  |
| --- |
| <%- css('style.css') %> // <link rel="stylesheet" href="/style.css">  <%- css(['style.css', 'screen.css']) %> // <link rel="stylesheet" href="/style.css"> // <link rel="stylesheet" href="/screen.css">  <%- css({ href: 'style.css', integrity: 'foo' }) %> // <link rel="stylesheet" href="/style.css" integrity="foo">  <%- css([{ href: 'style.css', integrity: 'foo' }, { href: 'screen.css', integrity: 'bar' }]) %> // <link rel="stylesheet" href="/style.css" integrity="foo"> // <link rel="stylesheet" href="/screen.css" integrity="bar"> |

#### js

载入 JavaScript 文件。path 可以是数组或字符串，如果 path 开头不是 / 或任何协议，则会自动加上根路径；如果后面没有加上 .js 扩展名的话，也会自动加上。Use object type for custom attributes.

|  |
| --- |
| <%- js(path, ...) %> |

**示例：**

|  |
| --- |
| <%- js('script.js') %> // <script src="/script.js"></script>  <%- js(['script.js', 'gallery.js']) %> // <script src="/script.js"></script> // <script src="/gallery.js"></script>  <%- js({ src: 'script.js', integrity: 'foo', async: true }) %> // <script src="/script.js" integrity="foo" async></script>  <%- js([{ src: 'script.js', integrity: 'foo' }, { src: 'gallery.js', integrity: 'bar' }]) %> // <script src="/script.js" integrity="foo"></script> // <script src="/gallery.js" integrity="bar"></script> |

#### link\_to

插入链接。

|  |
| --- |
| <%- link\_to(path, [text], [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| external | 在新视窗打开链接 | false |
| class | Class 名称 |  |
| id | ID |  |

**示例：**

|  |
| --- |
| <%- link\_to('http://www.google.com') %> // <a href="http://www.google.com" title="http://www.google.com">http://www.google.com</a>  <%- link\_to('http://www.google.com', 'Google') %> // <a href="http://www.google.com" title="Google">Google</a>  <%- link\_to('http://www.google.com', 'Google', {external: true}) %> // <a href="http://www.google.com" title="Google" target="\_blank" rel="noopener">Google</a> |

#### mail\_to

插入电子邮箱链接。

|  |
| --- |
| <%- mail\_to(path, [text], [options]) %> |

| **参数** | **描述** |
| --- | --- |
| class | Class 名称 |
| id | ID |
| subject | 邮件主题 |
| cc | 抄送（CC） |
| bcc | 密送（BCC） |
| body | 邮件内容 |

**示例：**

|  |
| --- |
| <%- mail\_to('a@abc.com') %> // <a href="mailto:a@abc.com" title="a@abc.com">a@abc.com</a>  <%- mail\_to('a@abc.com', 'Email') %> // <a href="mailto:a@abc.com" title="Email">Email</a> |

#### image\_tag

插入图片。

|  |
| --- |
| <%- image\_tag(path, [options]) %> |

| **参数** | **描述** |
| --- | --- |
| alt | 图片的替代文字 |
| class | Class 名称 |
| id | ID |
| width | 图片宽度 |
| height | 图片高度 |

#### favicon\_tag

插入 favicon。

|  |
| --- |
| <%- favicon\_tag(path) %> |

#### feed\_tag

插入 feed 链接。

|  |
| --- |
| <%- feed\_tag(path, [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| title | Feed 标题 |  |
| type | Feed 类型 | atom |

### 条件函数

#### is\_current

检查 path 是否符合目前页面的网址。开启 strict 选项启用严格比对。

|  |
| --- |
| <%- is\_current(path, [strict]) %> |

#### is\_home

检查当前页面是否为首页。

|  |
| --- |
| <%- is\_home() %> |

#### is\_post

检查当前页面是否为文章。

|  |
| --- |
| <%- is\_post() %> |

#### is\_page

检查当前页面是否为独立页面。

|  |
| --- |
| <%- is\_page() %> |

#### is\_archive

检查当前页面是否为存档页面。

|  |
| --- |
| <%- is\_archive() %> |

#### is\_year

检查当前页面是否为年度归档页面。

|  |
| --- |
| <%- is\_year() %> |

#### is\_month

检查当前页面是否为月度归档页面。

|  |
| --- |
| <%- is\_month() %> |

#### is\_category

检查当前页面是否为分类归档页面。  
如果给定一个字符串作为参数，将会检查目前是否为指定分类。

|  |
| --- |
| <%- is\_category() %> <%- is\_category('hobby') %> |

#### is\_tag

检查当前页面是否为标签归档页面。  
如果给定一个字符串作为参数，将会检查目前是否为指定标签。

|  |
| --- |
| <%- is\_tag() %> <%- is\_tag('hobby') %> |

### 字符串处理

#### trim

清除字符串开头和结尾的空格。

|  |
| --- |
| <%- trim(string) %> |

#### strip\_html

清除字符串中的 HTML 标签。

|  |
| --- |
| <%- strip\_html(string) %> |

**示例：**

|  |
| --- |
| <%- strip\_html('It\'s not <b>important</b> anymore!') %> // It's not important anymore! |

#### titlecase

把字符串转换为正确的 Title case。

|  |
| --- |
| <%- titlecase(string) %> |

**示例：**

|  |
| --- |
| <%- titlecase('this is an apple') %> # This is an Apple |

#### markdown

使用 Markdown 解析字符串。

|  |
| --- |
| <%- markdown(str) %> |

**示例：**

|  |
| --- |
| <%- markdown('make me \*\*strong\*\*') %> // make me <strong>strong</strong> |

#### render

解析字符串。

|  |
| --- |
| <%- render(str, engine, [options]) %> |

**Examples:**

|  |
| --- |
| <%- render('p(class="example") Test', 'pug'); %> // <p class="example">Test</p> |

See [Rendering](https://hexo.io/zh-cn/api/rendering) for more details.

#### word\_wrap

使每行的字符串长度不超过 length。length 预设为 80。

|  |
| --- |
| <%- word\_wrap(str, [length]) %> |

**示例：**

|  |
| --- |
| <%- word\_wrap('Once upon a time', 8) %> // Once upon\n a time |

#### truncate

移除超过 length 长度的字符串。length 的默认值是 30。

|  |
| --- |
| <%- truncate(text, length) %> |

**示例：**

|  |
| --- |
| <%- truncate('Once upon a time in a world far far away', {length: 17}) %> // Once upon a ti...  <%- truncate('Once upon a time in a world far far away', {length: 17, separator: ' '}) %> // Once upon a...  <%- truncate('And they found that many people were sleeping better.', {length: 25, omission: '... (continued)'}) %> // And they f... (continued) |

### 模板

#### partial

载入其他模板文件，您可在 locals 设定区域变量。

|  |
| --- |
| <%- partial(layout, [locals], [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| cache | 缓存（使用 Fragment cache） | false |
| only | 限制局部变量。在模板中只能使用 locals 中设定的变量。 | false |

#### fragment\_cache

局部缓存。它储存局部内容，下次使用时就能直接使用缓存。

|  |
| --- |
| <%- fragment\_cache(id, fn); |

**示例：**

|  |
| --- |
| <%- fragment\_cache('header', function(){  return '<header></header>'; }) %> |

### 日期与时间

#### date

插入格式化的日期。date 可以是 UNIX 时间、ISO 字符串、Date 对象或 [Moment.js](http://momentjs.com/) 对象。format 默认为 date\_format 配置信息。

|  |
| --- |
| <%- date(date, [format]) %> |

**示例：**

|  |
| --- |
| <%- date(Date.now()) %> // 2013-01-01  <%- date(Date.now(), 'YYYY/M/D') %> // Jan 1 2013 |

#### date\_xml

插入 XML 格式的日期。date 可以是 UNIX 时间、ISO 字符串、Date 对象或 [Moment.js](http://momentjs.com/) 对象。

|  |
| --- |
| <%- date\_xml(date) %> |

**示例：**

|  |
| --- |
| <%- date\_xml(Date.now()) %> // 2013-01-01T00:00:00.000Z |

#### time

插入格式化的时间。date 可以是 UNIX 时间、ISO 字符串、Date 对象或 [Moment.js](http://momentjs.com/) 对象。format 默认为 time\_format 配置信息。

|  |
| --- |
| <%- time(date, [format]) %> |

**示例：**

|  |
| --- |
| <%- time(Date.now()) %> // 13:05:12  <%- time(Date.now(), 'h:mm:ss a') %> // 1:05:12 pm |

#### full\_date

插入格式化的日期和时间。date 可以是 UNIX 时间、ISO 字符串、Date 对象或 [Moment.js](http://momentjs.com/) 对象。format默认为 date\_format + time\_format。

|  |
| --- |
| <%- full\_date(date, [format]) %> |

**示例：**

|  |
| --- |
| <%- full\_date(new Date()) %> // Jan 1, 2013 0:00:00  <%- full\_date(new Date(), 'dddd, MMMM Do YYYY, h:mm:ss a') %> // Tuesday, January 1st 2013, 12:00:00 am |

#### moment

[Moment.js](http://momentjs.com/) 函数库。

### 列表

#### list\_categories

插入分类列表。

|  |
| --- |
| <%- list\_categories([options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| orderby | 分类排列方式 | name |
| order | 分类排列顺序。1, asc 升序；-1, desc 降序。 | 1 |
| show\_count | 显示每个分类的文章总数 | true |
| style | 分类列表的显示方式。使用 list 以无序列表（unordered list）方式显示。 | list |
| separator | 分类间的分隔符号。只有在 style 不是 list 时有用。 | , |
| depth | 要显示的分类层级。0 显示所有层级的分类；-1 和 0 很类似，但是显示不分层级；1 只显示第一层的分类。 | 0 |
| class | 分类列表的 class 名称。 | category |
| transform | 改变分类名称显示方法的函数 |  |
| suffix | 为链接添加前缀 | None |

**用例:**

|  |
| --- |
| <%- list\_categories(post.categories, {  class: 'post-category',  transform(str) {  return titlecase(str);  } }) %>  <%- list\_categories(post.categories, {  class: 'post-category',  transform(str) {  return str.toUpperCase();  } }) %> |

#### list\_tags

插入标签列表。

|  |
| --- |
| <%- list\_tags([options]) %> |

| **选项** | **描述** | **预设值** |
| --- | --- | --- |
| orderby | 标签排列方式 | name |
| order | 标签排列顺序。1, asc 升序；-1, desc 降序。 | 1 |
| show\_count | 显示每个标签的文章总数 | true |
| style | 标签列表的显示方式。使用 list 以无序列表（unordered list）方式显示。 | list |
| separator | 标签间的分隔符号。只有在 style 不是 list 时有用。 | , |
| class | 标签列表的 class 名称。 | tag |
| transform | 改变标签名称显示方法的函数。请查看 [list\_categories](https://hexo.io/zh-cn/docs/helpers" \l "list-categories) 中给出的例子 |  |
| amount | 要显示的标签数量（0 = 无限制） | 0 |
| suffix | 为链接添加前缀 | None |

#### list\_archives

插入归档列表。

|  |
| --- |
| <%- list\_archives([options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| type | 类型。此设定可为 yearly 或 monthly。 | monthly |
| order | 排列顺序。1, asc 升序；-1, desc 降序。 | 1 |
| show\_count | 显示每个归档的文章总数 | true |
| format | 日期格式 | MMMM YYYY |
| style | 归档列表的显示方式。使用 list 以无序列表（unordered list）方式显示。 | list |
| separator | 归档间的分隔符号。只有在 style 不是 list 时有用。 | , |
| class | 归档列表的 class 名称。 | archive |
| transform | 改变归档名称显示方法的函数。请查看 [list\_categories](https://hexo.io/zh-cn/docs/helpers" \l "list-categories) 中给出的例子 |  |

#### list\_posts

插入文章列表。

|  |
| --- |
| <%- list\_posts([options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| orderby | 文章排列方式 | date |
| order | 文章排列顺序。1, asc 升序；-1, desc 降序。 | -1 |
| style | 文章列表的显示方式。使用 list 以无序列表（unordered list）方式显示。 | list |
| separator | 文章间的分隔符号。只有在 style 不是 list 时有用。 | , |
| class | 文章列表的 class 名称。 | post |
| amount | 要显示的文章数量（0 = 无限制） | 6 |
| transform | 改变文章名称显示方法的函数。请查看 [list\_categories](https://hexo.io/zh-cn/docs/helpers" \l "list-categories) 中给出的例子 |  |

#### tagcloud

插入标签云。

|  |
| --- |
| <%- tagcloud([tags], [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| min\_font | 最小字体尺寸 | 10 |
| max\_font | 最大字体尺寸 | 20 |
| unit | 字体尺寸的单位 | px |
| amount | 标签总量 | 40 |
| orderby | 标签排列方式 | name |
| order | 标签排列顺序。1, sac 升序；-1, desc 降序 | 1 |
| color | 使用颜色 | false |
| start\_color | 开始的颜色。您可使用十六进位值（#b700ff），rgba（rgba(183, 0, 255, 1)），hsla（hsla(283, 100%, 50%, 1)）或 [颜色关键字](http://www.w3.org/TR/css3-color/#svg-color)。此变量仅在 color 参数开启时才有用。 |  |
| end\_color | 结束的颜色。您可使用十六进位值（#b700ff），rgba（rgba(183, 0, 255, 1)），hsla（hsla(283, 100%, 50%, 1)）或 [颜色关键字](http://www.w3.org/TR/css3-color/#svg-color)。此变量仅在 color 参数开启时才有用。 |  |

### 其他

#### paginator

插入分页链接。

|  |
| --- |
| <%- paginator(options) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| base | 基础网址 | / |
| format | 网址格式 | page/%d/ |
| total | 分页总数 | 1 |
| current | 目前页数 | 0 |
| prev\_text | 上一页链接的文字。仅在 prev\_next 设定开启时才有用。 | Prev |
| next\_text | 下一页链接的文字。仅在 prev\_next 设定开启时才有用。 | Next |
| space | 空白文字 | … |
| prev\_next | 显示上一页和下一页的链接 | true |
| end\_size | 显示于两侧的页数 | 1 |
| mid\_size | 显示于中间的页数 | 2 |
| show\_all | 显示所有页数。如果开启此参数的话，end\_size 和 mid\_size 就没用了。 | false |
| escape | Escape HTML tags | true |

**Examples:**

|  |
| --- |
| <%- paginator({  prev\_text: '<',  next\_text: '>' }) %> |

|  |
| --- |
| <!-- Rendered as --> <a href="/1/">&lt;</a> <a href="/1/">1</a> 2 <a href="/3/">3</a> <a href="/3/">&gt;</a> |

|  |
| --- |
| <%- paginator({  prev\_text: '<i class="fa fa-angle-left"></i>',  next\_text: '<i class="fa fa-angle-right"></i>',  escape: false }) %> |

|  |
| --- |
| <!-- Rendered as --> <a href="/1/"><i class="fa fa-angle-left"></i></a> <a href="/1/">1</a> 2 <a href="/3/">3</a> <a href="/3/"><i class="fa fa-angle-right"></i></a> |

#### search\_form

插入 Google 搜索框。

|  |
| --- |
| <%- search\_form(options) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| class | 表单的 class name | search-form |
| text | 搜索提示文字 | Search |
| button | 显示搜索按钮。此参数可为布尔值（boolean）或字符串，当设定是字符串的时候，即为搜索按钮的文字。 | false |

#### number\_format

格式化数字。

|  |
| --- |
| <%- number\_format(number, [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| precision | 数字精度。此选项可为 false 或非负整数。 | false |
| delimiter | 千位数分隔符号 | , |
| separator | 整数和小数之间的分隔符号 | . |

**示例：**

|  |
| --- |
| <%- number\_format(12345.67, {precision: 1}) %> // 12,345.68  <%- number\_format(12345.67, {precision: 4}) %> // 12,345.6700  <%- number\_format(12345.67, {precision: 0}) %> // 12,345  <%- number\_format(12345.67, {delimiter: ''}) %> // 12345.67  <%- number\_format(12345.67, {separator: '/'}) %> // 12,345/67 |

#### meta\_generator

Inserts [generator tag](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta).

|  |
| --- |
| <%- meta\_generator() %> |

**Examples:**

|  |
| --- |
| <%- meta\_generator() %> // <meta name="generator" content="Hexo 4.0.0"> |

#### open\_graph

插入 open graph 资源。

|  |
| --- |
| <%- open\_graph([options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| title | 页面标题 (og:title) | page.title |
| type | 页面类型 (og:type) | blog |
| url | 页面网址 (og:url) | url |
| image | 页面图片 (og:image) | 内容中的图片 |
| site\_name | 网站名称 (og:site\_name) | config.title |
| description | 页面描述 (og:description) | 内容摘要或前 200 字 |
| twitter\_card | Twitter 卡片类型 (twitter:card) | summary |
| twitter\_id | Twitter ID (twitter:creator) |  |
| twitter\_site | Twitter 网站 (twitter:site) |  |
| google\_plus | Google+ 个人资料链接 |  |
| fb\_admins | Facebook 管理者 ID |  |
| fb\_app\_id | Facebook 应用程序 ID |  |

#### toc

解析内容中的标题标签 (h1~h6) 并插入目录。

|  |
| --- |
| <%- toc(str, [options]) %> |

| **参数** | **描述** | **默认值** |
| --- | --- | --- |
| class | Class 名称 | toc |
| list\_number | 显示编号 | true |
| max\_depth | 生成 TOC 的最大深度 | 6 |
| min\_depth | 生成 TOC 的最小深度 | 1 |

**示例：**

|  |
| --- |
| <%- toc(page.content) %> |

# Hexo插件

Hexo 有强大的插件系统，让我们能够轻松扩展功能而不用修改核心模块的源码。在 Hexo 中有两种形式的插件：

* 脚本（Scripts）
* 插件（Packages）

如果我们的代码很简单，我们可以编写脚本，只需要把 JavaScript 文件放到 scripts 文件夹，在启动时就会自动载入。简单来说，脚本文件可以相当于一些这样的的工具函数，当我们发现Hexo官方提供的函数不能满足我们的需求时，我们可以通过添加一个脚本来实现。

比如，我们现在有这样一个简单的需求，我们想给首页文章列表中的文章块添加一个背景颜色，背景颜色我们可以在文章md文件中定义，如果未定义，则随机选用一种颜色。

首先，我们先在文章md文件中顶部Front-matter添加一个color字段：

\_posts/hello-world-1.md:

title: Hello World 1  
date: 2019-02-12 17:49:32  
categories: 分类1  
tags:   
    - 标签1  
color: blue  
---

定义完成后，我们就可以在文章信息字段post或者page中获取到color。

然后，我们需要添加一个脚本函数来根据color字段来获取文章块的背景颜色，新增scripts/getPostBgColor.js:

const arr = [ 'blue', 'purple', 'green', 'yellow', 'red', 'orange' ];  
  
var getPostBgColor = function(color) {  
  if (arr.indexOf(color) >= 0) {  
    return `bg-${color}`;  
  }  
  return 'bg-' + randBgColor();  
};  
  
function randBgColor() {  
  return arr[randomInt(0, 5)];  
}  
  
function randomInt(min, max) {  
  return Math.round(Math.random() \* (max - min)) + min;  
}  
  
hexo.extend.helper.register('getPostBgColor', getPostBgColor);

最后一行我们通过hexo.extend.helper.register全局注册一个脚本函数，注册完成后，我们就可以在模板文件中和Hexo官方提供的全局函数一样使用。

修改layout/index.ejs:

...  
<article class="post <%= getPostBgColor(post.color) %>">  
...

添加背景颜色样式，编辑css/index.styl:

...  
.bg-blue {  
    background-color: #6fa3ef;  
}  
.bg-purple {  
    background-color: #bc99c4;  
}  
.bg-green {  
    background-color: #46c47c;  
}  
.bg-yellow {  
    background-color: #f9bb3c;  
}  
.bg-red {  
    background-color: #e8583d;  
}  
.bg-orange {  
    background-color: #f68e5f;  
}

看下效果，Hello World 1这篇文章我们定义了color：blue，因此是蓝色，其他文章，我们未定义color，因此是随机颜色。

这样，我们就实现了随机背景色这个小需求，当然这只是一个非常简单的例子，如果有其他复杂的需求，我们可以通过编写更加复杂的脚本来实现。

## 插件形式

Hexo 有强大的插件系统，使您能轻松扩展功能而不用修改核心模块的源码。在 Hexo 中有两种形式的插件：

### 脚本（Scripts）

如果您的代码很简单，建议您编写脚本，您只需要把 JavaScript 文件放到 scripts 文件夹，在启动时就会自动载入。

### 插件（Packages）

如果您的代码较复杂，或是您想要发布到 NPM 上，建议您编写插件。首先，在 node\_modules 文件夹中建立文件夹，文件夹名称开头必须为 hexo-，如此一来 Hexo 才会在启动时载入否则 Hexo 将会忽略它。

文件夹内至少要包含 2 个文件：一个是主程序，另一个是 package.json，描述插件的用途和所依赖的插件。

|  |
| --- |
| . ├── index.js └── package.json |

package.json 中至少要包含 name, version, main 属性，例如：

package.json

|  |
| --- |
| {  "name": "hexo-my-plugin",  "version": "0.0.1",  "main": "index" } |

### 工具

您可以使用 Hexo 提供的官方工具插件来加速开发：

* [hexo-fs](https://github.com/hexojs/hexo-fs)：文件 IO
* [hexo-util](https://github.com/hexojs/hexo-util)：工具程式
* [hexo-i18n](https://github.com/hexojs/hexo-i18n)：本地化（i18n）
* [hexo-pagination](https://github.com/hexojs/hexo-pagination)：生成分页资料

### 发布

当您完成插件后，可以考虑将它发布到 [插件列表](https://hexo.io/plugins)，让更多人能够使用您的插件。发布插件的步骤和 [更新文件](https://hexo.io/zh-cn/docs/contributing#%E6%9B%B4%E6%96%B0%E6%96%87%E4%BB%B6)非常类似。

1. Fork [hexojs/site](https://github.com/hexojs/site" \t "_blank)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

|  |
| --- |
| $ git clone https://github.com/<username>/site.git $ cd site $ npm install |

1. 编辑 source/\_data/plugins.yml，在档案中新增您的插件，例如：

|  |
| --- |
| - name: hexo-server  description: Server module for Hexo.  link: https://github.com/hexojs/hexo-server  tags:  - official  - server  - console |

1. 推送（push）分支。
2. 建立一个新的合并申请（pull request）并描述改动。

## 更新 Hexo 官方插件

我们也欢迎给 [Hexo 官方插件](https://github.com/hexojs" \t "_blank) 提交 PR 和 Issue 🤗

## 国际化（i18n）

若要让您的网站以不同语言呈现，您可使用国际化（internationalization）功能。请先在 \_config.yml 中调整 language 设定，这代表的是预设语言，您也可设定多个语言来调整预设语言的顺位。

|  |
| --- |
| language: zh-tw  language:  - zh-tw - en |

### 语言文件

语言文件可以使用 YAML 或 JSON 编写，并放在主题文件夹中的 languages 文件夹。您可以在语言文件中使用 [printf 格式](https://github.com/alexei/sprintf.js" \t "_blank)。

### 模板

在模板中，透过 \_\_ 或 \_p 辅助函数，即可取得翻译后的字符串，前者用于一般使用；而后者用于复数字符串。例如：

en.yml

|  |
| --- |
| index:  title: Home  add: Add  video:  zero: No videos  one: One video  other: %d videos |

|  |
| --- |
| <%= \_\_('index.title') %> // Home  <%= \_p('index.video', 3) %> // 3 videos |

### 路径

您可在 front-matter 中指定该页面的语言，也可在 \_config.yml 中修改 i18n\_dir 设定，让 Hexo 自动侦测。

|  |
| --- |
| i18n\_dir: :lang |

i18n\_dir 的预设值是 :lang，也就是说 Hexo 会捕获网址中的第一段以检测语言，举例来说：

|  |
| --- |
| /index.html => en /archives/index.html => en /zh-tw/index.html => zh-tw |

捕获到的字符串唯有在语言文件存在的情况下，才会被当作是语言，因此例二 /archives/index.html 中的 archives 就不被当成是语言。

### Hexo多语言化

#### 1.前言

大家都知道Hexo是一款很出色的生成静态网站的组件，对于初学者或量比较小的用户来说，静态网站托管可满足大部分需要。目前的托管平台有Github Page, Gitlab Page, 都大同小异，看自己的博客定位如果是要方便国外友人访问则最好是Github, 如果主要是面向国内用户则Gitlab也可以满足需求。

Hexo有很多Theme, 可在官网详细了解，适合不同风格的用户所需。对于本人来说，选型的标准主要是：简洁，大方，快速。基于此标准主要尝试的主题有:

* **maupassant**: 界面清爽，无多余内容，不过可惜对多语系博客支持不太好，只有简单的i18n国际化，不满足需求 ;
* **nova**: 尝试的另一款，作者很用心，把类似于博客园的风格借鉴到了hexo，号称支持双语系，不过尝试过程中发现切换语系无法切换中英博客，且两种语系博客无论怎么设置，都只能显示在一个列表中，所以也不满足需求;
* **minos**: 风格类似于maupassant, 比它强的是多语系支持这块比较完善，在尝试过后，正符合多语系的场景，遂采纳。

#### 2.前置条件

##### 2.1 环境准备

* **OS**: Darwin MacBook-Pro-2.local 17.7.0
* **NodeJS**: v10.10.0
* **Hexo**: 3.3.9
* **Github page**: 开通github账号

##### 2.2 软件安装

此步略，参考网上教程。安装好后，执行Hexo命令初始化一个静态网站目录:

hexo init hexo\_blog

##### 2.3 hexo依赖

**minos**: 从官方Github下下载

cd hexo\_blog

git clone https://github.com/ppoffice/hexo-theme-minos.git themes/minos

##### 2.4 npm依赖

* **hexo-renderer-sass**

cd hexo\_blog

npm install hexo-renderer-sass --save

* **cheerio**

cd hexo\_blog

npm install cheerio --save

#### 3.折腾过程

整个过程分为几个部分:

* **全站配置**: 配置站点
* **主题配置**: 配置minos
* **语系配置**: 配置切换语系

##### 3.1 全站配置

这里主要考虑和语系有关的配置，其它配置不在本文讨论范畴，大家可参照网上配置自由设定。全站配置文件为hexo\_blog/\_config.yml。配置过程如下：

###### 3.1.1 语系配置

#配置默认以English为主，中文为辅

language: [en,zh-cn]

###### 3.1.2 主题配置

#配置主题为minos

theme: minos

###### 3.1.3 url配置

#配置自己的静态网站部署地址

url: https://zendwind.github.io

###### 3.1.4 deploy配置

deploy:

#配置git部署模式

type: git

repo: https://github.com/zendwind/zendwind.github.io.git

branch: master

##### 3.2 主题配置

主题配置主要涉及语系方面，步骤如下:

###### 3.2.1 主题配置生成

cd test/themes/minos

#默认主题配置名称为\_config.yml.example

rsync -av \_config.yml.example \_config.yml

#生成英语系配置

rsync -av \_config.yml.example \_config.en.yml

#生成中文语系配置

rsync -av \_config.yml.example \_config.zh-cn.yml

###### 3.2.2 多语系配置

* \_英文语系配置: config.en.yml

menu:

Archives: /archives

Categories: /categories

Tags: /tags

About: /about

* \_中文语系配置: config.zh-cn.yml

menu:

归档: /zh-cn/zh-cn/archives #这里是个bug, 如果不加双层zh-cn后面无法定位到archives目录

分类: /zh-cn/categories

关于: /zh-cn/about

##### 3.3 站点文档配置

上面已经把hexo配置完成了，接下来为了要实现双语博客的访问，需要配置文档目录。首先，我们来看下没配置之前的站点目录结构是怎么样的：

.

source

└── \_posts

└── hello-world.md

在站点根目录下，有一个source目录，同时里面又有一个\_post子目录，初始化时会生成一个hello-world.md的文档。由于我们站点的默认语系是English, 所以在\_post目录下存放的就是英文文档。

###### 3.3.1 source/\_post目录文档配置

在source/\_posts目录下，新增一个zh-cn的目录，专门用于存放中文文档。在zh-cn目录下新增一个中文文档命名为hello-wolrd-cn.md。此时目录结构如下:

.

source

└── \_posts

├── hello-world.md

└── zh-cn

└── hello-world-cn.md

###### 3.3.2 source目录文档配置

在\_post增加完中文文档目录后，在source目录直接再新增一个zh-cn目录，用于存放其它一些中文文档,　如about之类的。同时在source目录下新增一个about目录用于存放英文的about文档，目录结构如下:

source

├── \_posts

│   ├── hello-world.md

│   └── zh-cn

│   └── hello-world-cn.md

├── about

│   └── index.md

└── zh-cn

└── about

└── index.md

关于about中的index.md, 其内容如下:

* **英文语系**

---

layout: "about"

title: 'About Me'

date: 2020-04-24 10:00:00

---

* **中文语系**

---

layout: "about"

title: '关于我'

date: 2020-04-26 10:00:00

--

测试About页面

Test About Page  
这样初步完成站点文档的目录设置。

###### 3.3.3 增加多语系切换配置

此步，需要修改下脚本，把语系相关代码调整到合适位置，如在themes/minos/layout/common/footer.ejs脚本中，找到:

<%- partial('common/languages') %>

把上述代码剪切到themes/minos/layout/common/navbar.ejs的合适位置（75行）,　如下：

<% } %>

<%- partial('common/languages') %>

</div>

</div>

</nav>

同时找到themes/minos/layout/common/languages.ejs, 增加一个配置项style="top:100%" 到如下代码中:

<div class="dropdown-menu has-text-left" role="menu" style="top:100%">

##### 3.4 查看效果

###### 3.4.1 编译静态文件

cd hexo\_blog

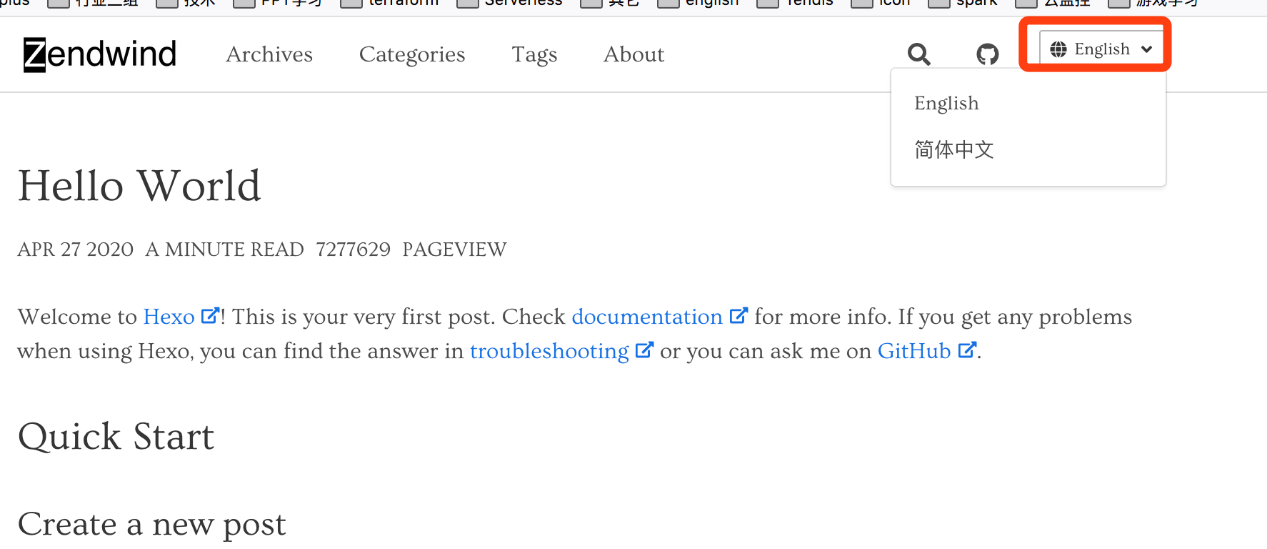
hexo g

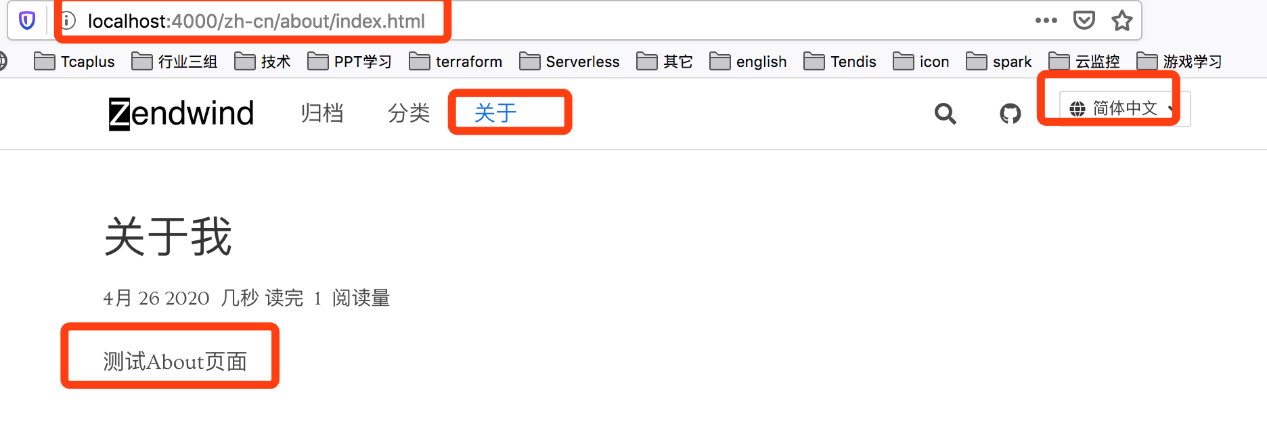
###### 3.4.2 本地测试

cd hexo\_blog

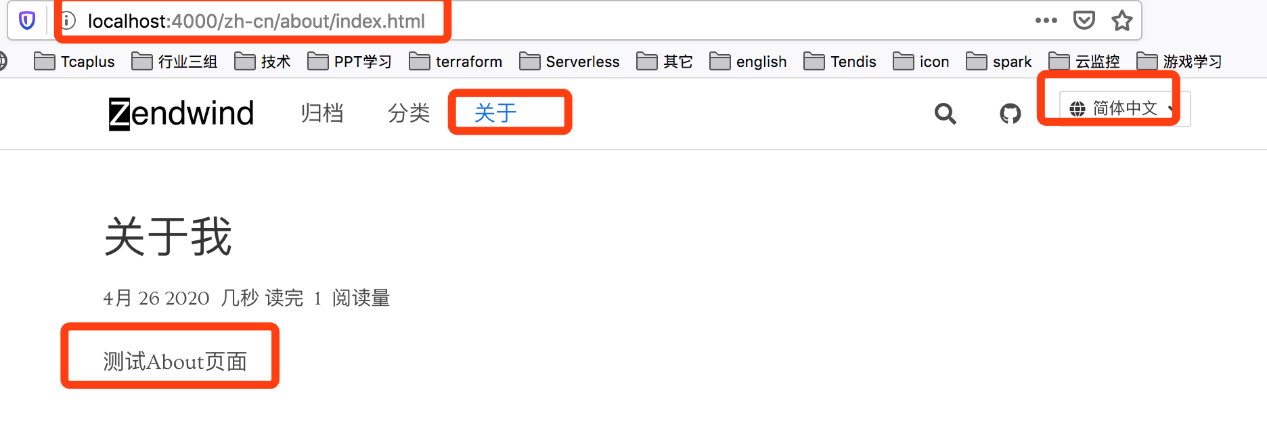
hexo s

打开浏览器，查看localhost:4000网页是否正常。

打开默认的英文语系站点测试效果如下:  




切换到中文语系站点测试效果如下:  

##### 3.5 部署到github

如果查看效果ok, 可以把站点部署到github上，执行命令如下:

hexo d -g

备注：在部署之前，需要检查是否有创建好站点仓库（在配置中指定的repo），　如https://github.com/zendwind/zendwind.github.io.git。

然后通过zendwind.github.io直接访问站点，看是否ok。

#### 4. 其它功能

如果要为你博客添加评论系统、添加计数及分享功能，在hexo中也比较方便。

#### 4.1 添加评论功能

评论系统有很多，主流的有两种：

* **国外**: Disqus, 这个在国内访问很慢
* **国内**: Valine, 这个在国内访问比较快  
  本文采用Valine模式来设置评论功能，Valine已和LeanCloud打通，即评论数据可放在LeanCloud存储，且是免费的，方便广大初学者学习使用。设置步骤如下：

###### 4.1.1 创建应用

登录leancloud.com官网，注册账号,　登录后创建一个开发版应用，如下图所示:  


###### 4.1.2 查看应用app\_id和app\_key



###### 4.1.3 配置应用key

在主题配置中配置comment：\_config.en.yml 和\_config.zh-cn.yml

comments:

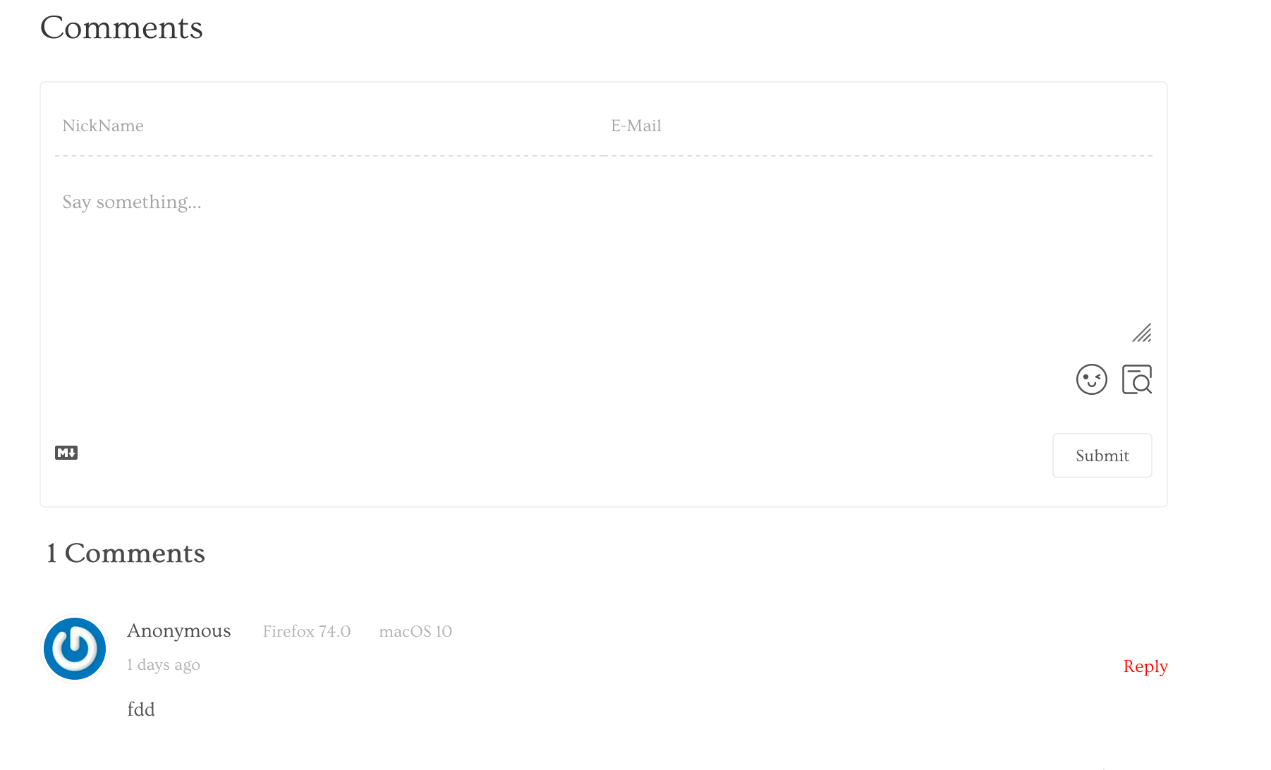
type: valine

app\_id: xxx #replace your app id

app\_key: xxx # replace your app key

###### 4.1.4 文章开启comment或关闭comment

在文档开头可以通过设置comments: false来关闭文章评论

评论效果类似如下:  


##### 4.2 添加文章访问量及站点访问统计功能

一般有两种模式：

* **Valine模式**: 集成Vistor统计功能，与leancloud打通，把统计数据存储到leancloud
* **不蒜子模式**: 简单，直接集成脚本即可。

本文主要介绍用不蒜子模式来统计访问量，步骤如下：

###### 4.2.1 修改主题配置

修改主题配置 ：\_config.en.yml和\_config.zh-cn.yml，增加如下内容：

article:

#增加一个pageview变量，用来控制是否显示文章访问量

pageview: true

###### 4.2.2 增加busuanzi脚本

在主题目录下，找到themes/minos/layout/common/head.ejs, 在最后面添加如下代码：

<% if (!has\_config('article.pageview') || get\_config('article.pageview') === true) { %>

<script async="" src="//busuanzi.ibruce.info/busuanzi/2.3/busuanzi.pure.mini.js"></script>

<% } %>

###### 4.2.3 增加访问量统计

在主题目录下，找到themes/minos/layout/common/article.ejs, 在合适位置(第28行后），添加如下代码：

<% if (!has\_config('article.pageview') || get\_config('article.pageview') === true) { %>

<span class="column is-narrow" id="busuanzi\_value\_page\_pv">0</span>

<span class="column is-narror"> <%=`${ \_\_('article.vistor') }` %></span>

<% } %>

###### 4.2.4 修改多语系配置

在themes/minos/languages目录下，分别修改en.yml和zh-cn.yml,增加两个对应项，如下所示:

* **en.yml**

article:

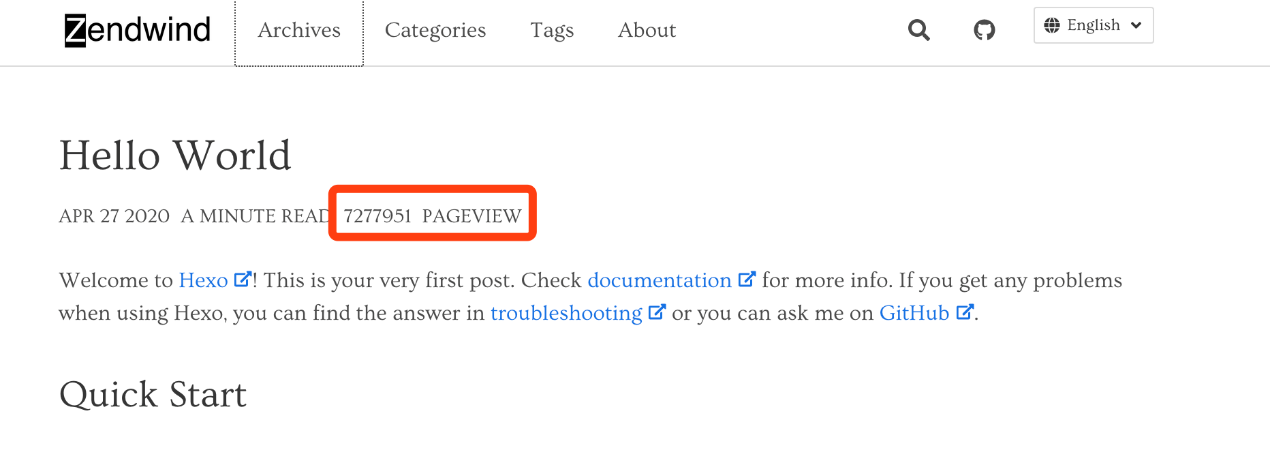
vistor: PageView

* **zh-cn.yml**

article:

vistor: 阅读量

###### 4.2.5 查看效果

添加完后，类似如下截图显示，文章访问量：  


#### 5.总结

本文主要描述了关于Hexo如何搭建多语系方面的内容，同时结合很常见的一些功能展开了介绍，希望大家能从本文中获得一些帮助。

### Hexo的多语言配置

关于**Hexo**的中英文语言配置，我的做法是，在菜单栏添加一个按键，通过按键切换到中/英文界面，让中文文章只显示在中文页面，英文显示在英文页面。

#### 插件安装

首先，安装**hexo**-generator-i18n插件

|  |  |
| --- | --- |
| 1 | npm install **hexo**-generator-i18n --save |

然后修改\_config.yml

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | language: [zh-CN,en] # **hexo**-generator-i18n 选项（可选，默认使用如下设置） i18n:  type: [page, post]  generator: [index, archive, category, tag] |

这里并没有明白tyte和generator的作用，但是似乎不影响

之后是**在主题languages目录下添加对应的语言.yml文件(如zh.yml, en.yml)**

但是由于我使用的主题并不支持**多**语言，我也懒得弄，所有暂时没有管这一项

插件安装完成后，执行**hexo** clean & **hexo** g

会生成public/en目录，对于\_post内的所有博文，都会在public里存放一份，en下存放一份。

那么就需要根据页面的语言，显示不同的博文。

#### 博文配置

对于所有的markdown文章，都需要在文章内添加属性lang:zh-CN or lang:en，来区分中英文

#### 网页配置

我通过url来区分中英文页面，对于中文，url是www.yoursite.com，英文，url是www.yoursite.com/en/,

这样，在js中检索url是否包含en，来判断显示中文还是英文界面。

**修改主题**, 我的主题路径是themes/vexo/layout/index.ejs

获取当前html的路径，并判断路径内是否存在en

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 | <main class="app-body">  <% var currentURL = page.path.split('/') %>  <% if (currentURL.length === 1) { %>  <% var lang = page.lang %>  <% } else { %>  <% var lang = currentURL.shift() %>  <% } %>  <% page.posts.each(function(post) { %>  <% if (lang == post.lang) { %>  <article class="article-card">  <h2 class="article-head">  <a href="<%- url\_for(post.path) %>"><%- post.title %></a>  </h2>  <p class="article-date"><%- date(post.date, "LL") %></p>  <% if (post.tags && post.tags.length) { %>  <%- partial('\_partial/tag', { tags: post.tags }) %>  <% } %>  <div class="article-summary">  <% if (post.excerpt) { %>  <%- post.excerpt %>  <% } else { %>  <%- truncate(strip\_html(post.content), { length: 150, omission: ' ...' }) %>  <% } %>  </div>  <a class="more" href="<%- url\_for(post.path) %>"><%- theme.excerpt\_link %></a>  </article>  <% } %>  <% }) %>   <% if (page.total > 1){ %>  <%- partial('\_partial/pager') %>  <% } %> </main> |

上述都完成，就可以通过www.yoursite.com/en进入英文网页，且该网页只有英文文章

#### 添加切换语言按钮

修改theme下的\_config.yml文件, 添加一个EN的按钮，跳转到英文界面

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | menu:  Home: /  Tags: /tags/  Archives: /archives/  Projects: /project/  About: /about/  EN: /en/ |

这里为了简单，没有动态修改EN这个按钮，提供从英文网页，跳转回中文网页的方式(实际上，点击Home按键，就跳转回中文了)

#### 部署测试

|  |  |
| --- | --- |
| 1 2 | **hexo** clean & **hexo** g **hexo** d |

##### 

##### 

##### 

## hexo-abbrlink

### How to install

Add plugin to Hexo:

npm install hexo-abbrlink --save

Modify permalink in config.yml file:

permalink: posts/:abbrlink/

There are two settings:

alg -- Algorithm (currently support crc16 and crc32, which crc16 is default)

rep -- Represent (the generated link could be presented in hex or dec value)

# abbrlink config

abbrlink:

alg: crc32 #support crc16(default) and crc32

rep: hex #support dec(default) and hex

drafts: false #(true)Process draft,(false)Do not process draft. false(default)

# Generate categories from directory-tree

# depth: the max\_depth of directory-tree you want to generate, should > 0

auto\_category:

enable: true #true(default)

depth: #3(default)

auto\_title: false #enable auto title, it can auto fill the title by path

auto\_date: false #enable auto date, it can auto fill the date by time today

force: false #enable force mode,in this mode, the plugin will ignore the cache, and calc the abbrlink for every post even it already had abbrlink.

### Sample

The generated link will look like the following:

crc16 & hex

https://post.zz173.com/posts/66c8.html

crc16 & dec

https://post.zz173.com/posts/65535.html

crc32 & hex

https://post.zz173.com/posts/8ddf18fb.html

crc32 & dec

https://post.zz173.com/posts/1690090958.html

### Limitation

[fixed] Maximum number of posts is 65535 for crc16. (now, if a abbrlink already exist, it will change another one and try again and again...)

### More info

see [this](https://post.zz173.com/detail/hexo-abbrlink.html)(Chinese)

### Hexo 巧用 abbrlink 插件实现文章多语言版本

　　目前，Hexo 似乎没有完美的实现文章多语言版本的方式，而有时针对某些特定的文章，有多语言版本的需求，而呉真的博客又使用 [hexo-abbrlink](https://github.com/rozbo/hexo-abbrlink" \t "_blank) 插件实现持久连接的生成。基于这些条件，这里提出一种利用 hexo-abbrlink 插件生成多语言文章的方法。

#### 使用方法

　　为每一种语言生成一个 md 文件，可以命名为 xxx.md，xxx\_en.md，xxx\_jp.md 格式，或者将额外语言的文件放入资源文件夹内，例如 xxx/en.md，xxx/jp.md。在 hexo-abbrlink 插件生成永久链接后，手动将链接修改为 89abcdef\_en 之类的格式，如此你即可生成多个文章页面，每个页面对应一种语言版本，你可以在每个语言版本中自行添加超链接至别的语言页面。

#### 在首页中隐藏不需要的语言

　　利用这个方法可以针对有多重语言需求的文章灵活的撰写多语言版本，但是同样也有缺点，在首页中以及分类菜单下，会同时出现同篇文章的多个语言版本，再利用 [hexo-hide-posts](https://github.com/printempw/hexo-hide-posts" \t "_blank) 插件，将不需要显示在首页上的语言版本隐藏。

提示：你可以在配置文件内设置在 sitemap 中不隐藏文章，利于搜索引擎收录多语言版本。  
演示文章：[修改群晖存储池及存储空间顺序](https://blog.kuretru.com/posts/2752cb63/)

## hexo-hide-posts

[**English Documentation**](https://github.com/printempw/hexo-hide-posts/blob/master/README.md)

本 Hexo 插件可以在博客中隐藏指定的文章，并使它们仅可通过链接访问。

当一篇文章被设置为「隐藏」时，它不会出现在任何列表中（包括首页、存档、分类页面、标签页面、Feed、站点地图等），也不会被搜索引擎索引（前提是搜索引擎遵守 noindex 标签）。

只有知道文章链接的人才可以访问被隐藏的文章。

### 安装

$ npm install hexo-hide-posts --save

### 使用

在文章的 [front-matter](https://hexo.io/docs/front-matter) 中添加 hidden: true 即可隐藏文章。

比如我们隐藏了 source/\_posts/lorem-ipsum.md 这篇文章：

---

title: 'Lorem Ipsum'

date: '2019/8/10 11:45:14'

hidden: true

---

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

虽然首页上被隐藏了，但你仍然可以通过 https://hexo.test/lorem-ipsum/ 链接访问它。

你可以在命令行运行 hexo hidden:list 来获取当前所有的已隐藏文章列表。

插件也在 [Local Variables](https://hexo.io/api/locals) 中添加了 all\_posts 和 hidden\_posts 变量，供自定义主题使用。

### 配置

在你的站点 \_config.yml 中添加如下配置：

# hexo-hide-posts

hide\_posts:

# 可以改成其他你喜欢的名字

filter: hidden

# 指定你想要传递隐藏文章的 generator，比如让所有隐藏文章在存档页面可见

# 常见的 generators 有：index, tag, category, archive, sitemap, feed, etc.

public\_generators: []

# 为隐藏的文章添加 noindex meta 标签，阻止搜索引擎收录

noindex: true

举个栗子：设置 filter: secret 之后，你就可以在 front-matter 中使用 secret: true 来隐藏文章了

## hexo-tag-aplayer

### 安装

npm install --save hexo-tag-aplayer

### 依赖

* APlayer.js > 1.8.0
* Meting.js > 1.1.1

### 使用

{% aplayer title author url [picture\_url, narrow, autoplay, width:xxx, lrc:xxx] %}

#### 标签参数

* title : 曲目标题
* author: 曲目作者
* url: 音乐文件 URL 地址
* picture\_url: (可选) 音乐对应的图片地址
* narrow: （可选）播放器袖珍风格
* autoplay: (可选) 自动播放，移动端浏览器暂时不支持此功能
* width:xxx: (可选) 播放器宽度 (默认: 100%)
* lrc:xxx: （可选）歌词文件 URL 地址

当开启 Hexo 的 [文章资源文件夹](https://hexo.io/zh-cn/docs/asset-folders.html#%E6%96%87%E7%AB%A0%E8%B5%84%E6%BA%90%E6%96%87%E4%BB%B6%E5%A4%B9) 功能时，可以将图片、音乐文件、歌词文件放入与文章对应的资源文件夹中，然后直接引用：

{% aplayer "Caffeine" "Jeff Williams" "caffeine.mp3" "picture.jpg" "lrc:caffeine.txt" %}

#### 歌词标签

除了使用标签 lrc 选项来设定歌词，你也可以直接使用 aplayerlrc 标签来直接插入歌词文本在博客中：

{% aplayerlrc "title" "author" "url" "autoplay" %}

[00:00.00]lrc here

{% endaplayerlrc %}

#### 播放列表

{% aplayerlist %}

{

"narrow": false, // （可选）播放器袖珍风格

"autoplay": true, // （可选) 自动播放，移动端浏览器暂时不支持此功能

"mode": "random", // （可选）曲目循环类型，有 'random'（随机播放）, 'single' (单曲播放), 'circulation' (循环播放), 'order' (列表播放)， 默认：'circulation'

"showlrc": 3, // （可选）歌词显示配置项，可选项有：1,2,3

"mutex": true, // （可选）该选项开启时，如果同页面有其他 aplayer 播放，该播放器会暂停

"theme": "#e6d0b2", // （可选）播放器风格色彩设置，默认：#b7daff

"preload": "metadata", // （可选）音乐文件预载入模式，可选项： 'none' 'metadata' 'auto', 默认: 'auto'

"listmaxheight": "513px", // (可选) 该播放列表的最大长度

"music": [

{

"title": "CoCo",

"author": "Jeff Williams",

"url": "caffeine.mp3",

"pic": "caffeine.jpeg",

"lrc": "caffeine.txt"

},

{

"title": "アイロニ",

"author": "鹿乃",

"url": "irony.mp3",

"pic": "irony.jpg"

}

]

}

{% endaplayerlist %}

#### MeingJS 支持 (3.0 新功能)

[MetingJS](https://github.com/metowolf/MetingJS) 是基于[Meting API](https://github.com/metowolf/Meting) 的 APlayer 衍生播放器，引入 MetingJS 后，播放器将支持对于 QQ音乐、网易云音乐、虾米、酷狗、百度等平台的音乐播放。

如果想在本插件中使用 MetingJS，请在 Hexo 配置文件 \_config.yml 中设置：

aplayer:

meting: true

接着就可以通过 {% meting ...%} 在文章中使用 MetingJS 播放器了：

<!-- 简单示例 (id, server, type) -->

{% meting "60198" "netease" "playlist" %}

<!-- 进阶示例 -->

{% meting "60198" "netease" "playlist" "autoplay" "mutex:false" "listmaxheight:340px" "preload:none" "theme:#ad7a86"%}

有关 {% meting %} 的选项列表如下:

| **选项** | **默认值** | **描述** |
| --- | --- | --- |
| id | **必须值** | 歌曲 id / 播放列表 id / 相册 id / 搜索关键字 |
| server | **必须值** | 音乐平台: netease, tencent, kugou, xiami, baidu |
| type | **必须值** | song, playlist, album, search, artist |
| fixed | false | 开启固定模式 |
| mini | false | 开启迷你模式 |
| loop | all | 列表循环模式：all, one,none |
| order | list | 列表播放模式： list, random |
| volume | 0.7 | 播放器音量 |
| lrctype | 0 | 歌词格式类型 |
| listfolded | false | 指定音乐播放列表是否折叠 |
| storagename | metingjs | LocalStorage 中存储播放器设定的键名 |
| autoplay | true | 自动播放，移动端浏览器暂时不支持此功能 |
| mutex | true | 该选项开启时，如果同页面有其他 aplayer 播放，该播放器会暂停 |
| listmaxheight | 340px | 播放列表的最大长度 |
| preload | auto | 音乐文件预载入模式，可选项： none, metadata, auto |
| theme | #ad7a86 | 播放器风格色彩设置 |

关于如何设置自建的 Meting API 服务器地址，以及其他 MetingJS 配置，请参考章节[自定义配置](https://github.com/MoePlayer/hexo-tag-aplayer/blob/master/docs/README-zh_cn.md" \l "%E8%87%AA%E5%AE%9A%E4%B9%89%E9%85%8D%E7%BD%AE30-%E6%96%B0%E5%8A%9F%E8%83%BD)

#### PJAX 兼容

若在 Hexo 中使用了 PJAX，可能需要自己手动清理 APlayer 全局实例：

$(document).on('pjax:start', function () {

if (window.aplayers) {

for (let i = 0; i < window.aplayers.length; i++) {

window.aplayers[i].destroy();

}

window.aplayers = [];

}

});

### 自定义配置（3.0 新功能）

现在你可以在 Hexo 配置文件 \_config.yml 中配置本插件：

aplayer:

script\_dir: some/place # Public 目录下脚本目录路径，默认: 'assets/js'

style\_dir: some/place # Public 目录下样式目录路径，默认: 'assets/css'

cdn: http://xxx/aplayer.min.js # 引用 APlayer.js 外部 CDN 地址 (默认不开启)

style\_cdn: http://xxx/aplayer.min.css # 引用 APlayer.css 外部 CDN 地址 (默认不开启)

meting: true # MetingJS 支持

meting\_api: http://xxx/api.php # 自定义 Meting API 地址

meting\_cdn: http://xxx/Meing.min.js # 引用 Meting.js 外部 CDN 地址 (默认不开启)

asset\_inject: true # 自动插入 Aplayer.js 与 Meting.js 资源脚本, 默认开启

externalLink: http://xxx/aplayer.min.js # 老版本参数，功能与参数 cdn 相同

### 故障排除

#### 标签参数空格问题

在 Hexo 标签中，用户可能无法直接在标签参数中[加入空格](https://github.com/hexojs/hexo/issues/1455)

如果遇到这类问题，请直接将参数用双引号括起来使用，如下所示：

{% aplayer "Caffeine" "Jeff Williams" "caffeine.mp3" "autoplay" "width:70%" "lrc:caffeine.txt" %}

#### 重复载入 Aplayer.js 资源脚本问题

本插件通过 after\_render:html过滤器 , 将 APlayer.js 和 Meting.js 插入到使用了本插件标签 的 HTML 文件中:

<html>

<head>

...

<script src="assets/js/aplayer.min.js"></script>

<script src="assets/js/meting.min.js"></script>

</head>

...

</html>

但是 after\_render:html 在一些情形下可能无法被正常触发:

* [Does not work with hexo-renderer-jade](https://github.com/hexojs/hexo-inject/issues/1)
* after\_render:html 似乎在 Hexo 服务器模式默认配置中无法被调用 (hexo server), 遇到这种情况用户可能需要使用 hexo-server 的静态文件解析模式 ( hexo server -s) .

如果在博客生成过程中，插件发现 after\_render:html 没有被调用，那么插件将会通过 after\_post\_render 过滤器来植入脚本。但是使用 after\_post\_render 会有重复载入 APlayer.js 的情况（例如当一个页面中存在多篇博客时），以及一些非文章页面将无法使用本插件。

如果想完全解决这个问题，用户可能需要自己在主题文件中手动加入 Aplayer.js 与 Meting.js，同时关闭插件的自动脚本插入功能：

aplayer:

asset\_inject: false

## hexo-douban

### 安装

$ npm install hexo-douban --save

### 配置

将下面的配置写入站点的配置文件 \_config.yml 里(不是主题的配置文件).

douban:

user: mythsman

builtin: false

book:

title: 'This is my book title'

quote: 'This is my book quote'

movie:

title: 'This is my movie title'

quote: 'This is my movie quote'

game:

title: 'This is my game title'

quote: 'This is my game quote'

timeout: 10000

* **user**: 你的豆瓣ID.打开豆瓣，登入账户，然后在右上角点击 "个人主页" ，这时候地址栏的URL大概是这样："<https://www.douban.com/people/xxxxxx/>" ，其中的"xxxxxx"就是你的个人ID了。
* **builtin**: 是否将生成页面的功能嵌入hexo s和hexo g中，默认是false,另一可选项为true(1.x.x版本新增配置项)。
* **title**: 该页面的标题.
* **quote**: 写在页面开头的一段话,支持html语法.
* **timeout**: 爬取数据的超时时间，默认是 10000ms ,如果在使用时发现报了超时的错(ETIMEOUT)可以把这个数据设置的大一点。

如果只想显示某一个页面(比如movie)，那就把其他的配置项注释掉即可。

### 使用

#### 1.x.x版本

在0.x.x版本中，文章的更新和豆瓣页面的爬取操作是绑定在一起的，无法支持单独更新文章或者单独爬取文章。

在1.x.x版本中，使用hexo douban命令即可生成指定页面，如果不加参数，那么默认参数为-bgm。

**需要注意的是**，通常大家都喜欢用hexo d来作为hexo deploy命令的简化，但是当安装了hexo douban之后，就不能用hexo d了，因为hexo douban跟hexo deploy的前缀都是hexo d。

$ hexo douban -h

Usage: hexo douban

Description:

Generate pages from douban

Options:

-b, --books Generate douban books only

-g, --games Generate douban games only

-m, --movies Generate douban movies only

如果配置了builtin参数为true，那么除了可以使用hexo douban命令之外，hexo g或hexo s也内嵌了生成页面的功能。

#### 0.x.x版本

直接使用命令hexo g即在生成静态页面前爬取豆瓣数据，如果使用hexo s则会监听文件变动，每有一次变动就会重新爬取数据。

### 升级

我会不定期更新一些功能或者修改一些Bug，所以如果想使用最新的特性，可以用下面的方法来更新:

1. 修改package.json内hexo-douban的版本号至最新
2. 重新安装最新版本npm install hexo-douban --save

或者使用npm install hexo-douban --update --save直接更新。

### 显示

如果上面的配置和操作都没问题，就可以在生成站点之后打开 //yourblog/books 和 //yourblog/movies, //yourblog/games, 来查看结果.

### 菜单

如果上面的显示没有问题就可以在主题的配置文件 \_config.yml 里添加如下配置来为这些页面添加菜单链接.

menu:

Home: /

Archives: /archives

Books: /books #This is your books page

Movies: /movies #This is your movies page

Games: /games #This is your games page

## hexo-generator-indexed

### Installation

$ npm install hexo-generator-indexed

### Options

Add or modify the following section to your root \_config.yml file

index\_generator:

path: ''

per\_page: 10

order\_by: -date

pagination\_dir: page

* **path**: Root path for your blog's index page.
  + default: ""
* **per\_page**: Posts displayed per page.
  + default: [config.per\_page](https://hexo.io/docs/configuration.html" \l "Pagination) as specified in the official Hexo docs (if present), otherwise 10
  + 0 disables pagination
* **order\_by**: Posts order.
  + default: date descending
* **pagination\_dir**: URL format.
  + default: 'page'
  + awesome-page makes the URL ends with 'awesome-page/' for second page and beyond.

### Note

If your theme define a non-archive index layout (e.g. About Me page), this plugin would follow that layout instead and not generate an archive. In that case, use [hexo-generator-archive](https://github.com/hexojs/hexo-generator-archive) to generate an archive according to the archive layout.

### Feature & Usage

## 置顶文章 Pinned article

在文章的 [Front-matter](https://hexo.io/docs/front-matter) 中增加一个 sticky 参数用来置顶，其值应为大于0的整数，表示置顶的优先级（未指定则默认为 0）。数字越大，文章越靠前。  
Add a sticky parameter to the the post's [Front-matter](https://hexo.io/docs/front-matter) to make the post pinned at the top of home page.

---

title: example

sticky: 100

---

采用 sticky 关键词是因为这个在 [NexT 主题](https://github.com/next-theme/hexo-theme-next)中已经定义了置顶样式，因此可以直接拿来使用。

## 隐藏文章 Hide article

在文章的 Front-matter 中增加一个 hide 参数用来隐藏。  
Add a hide parameter to the the post's Front-matter to hide the post from home page.

---

title: example

hide: true

---

## 隐藏特定分类中的文章 Hide articles in specified categories

在 Hexo 的 \_config.yml 中可以通过 hide\_categories 选项设置隐藏某个分类下的文章，例如：  
You can hide articles in specified categories by setting hide\_categories in hexo \_config.yml, e.g.

hide\_categories:

- categorie1

- categorie2

## 新插件预留

## 插件使用的问题解答

在使用 Hexo 时，您可能会遇到一些问题，下列的常见问题解答可能会对您有所帮助。如果您在这里找不到解答，可以在 [GitHub](https://github.com/hexojs/hexo/issues) 或 [Google Group](https://groups.google.com/group/hexo) 上提问。

### YAML 解析错误

|  |
| --- |
| JS-YAML: incomplete explicit mapping pair; a key node is missed at line 18, column 29:  last\_updated: Last updated: %s |

如果 YAML 字符串中包含冒号（:）的话，请加上引号。

|  |
| --- |
| JS-YAML: bad indentation of a mapping entry at line 18, column 31:  last\_updated:"Last updated: %s" |

请确认您使用空格进行缩进（Soft tab），并确认冒号后有加上一个空格。

您可参阅 [YAML 规范](http://www.yaml.org/spec/1.2/spec.html) 以取得更多信息。

### EMFILE 错误

|  |
| --- |
| Error: EMFILE, too many open files |

虽然 Node.js 有非阻塞 I/O，同步 I/O 的数量仍被系统所限制，在生成大量静态文件的时候，您可能会碰到 EMFILE 错误，您可以尝试提高同步 I/O 的限制数量来解决此问题。

|  |
| --- |
| $ ulimit -n 10000 |

（这一命令只对Linux系统有效）

### Git 部署问题

#### RPC failed

|  |
| --- |
| error: RPC failed; result=22, HTTP code = 403  fatal: 'username.github.io' does not appear to be a git repository |

请确认您已经在电脑上 [配置 git](https://help.github.com/articles/set-up-git)，或改用 HTTPS 库（repository）地址。

#### Error: ENOENT: no such file or directory

如果你遇到了这个错误，有可能是你的文件名、分类或者标签的名字混淆了大写和小写，你可以尝试检查每一个标签和分类的名称，是否大小写一致来修复这一问题。

### 服务器问题

|  |
| --- |
| Error: listen EADDRINUSE |

您可能同时开启两个 Hexo 服务器，或者有其他应用程序正在占用相同的端口，请尝试修改 port 参数，或是在启动 Hexo 服务器时加上 -p 选项。

|  |
| --- |
| $ hexo server -p 5000 |

### 插件安装问题

|  |
| --- |
| npm ERR! node-waf configure build |

当您尝试安装以 C/C++ 或其他非 JavaScript 语言所编写的插件时，可能会遇到此类问题，请确认您已经在电脑上安装相对应的编译器。

### DTrace 错误 （Mac OS X）

|  |
| --- |
| { [Error: Cannot find module './build/Release/DTraceProviderBindings'] code: 'MODULE\_NOT\_FOUND' } { [Error: Cannot find module './build/default/DTraceProviderBindings'] code: 'MODULE\_NOT\_FOUND' } { [Error: Cannot find module './build/Debug/DTraceProviderBindings'] code: 'MODULE\_NOT\_FOUND' } |

DTrace 安装可能有错误 , 使用下列命令:

|  |
| --- |
| $ npm install hexo --no-optional |

参考 [#1326](https://github.com/hexojs/hexo/issues/1326#issuecomment-113871796)

### 在 Jade 或 Swig 遍历资料

Hexo 使用 [Warehouse](https://github.com/hexojs/warehouse) 存储资料，它不是一般数组所以必须先进行类型转型才能遍历。

|  |
| --- |
| {% for post in site.posts.toArray() %} {% endfor %} |

### 资料没有更新

有时资料可能没有被更新，或是生成的文件与修改前的相同，您可以尝试清除缓存并再执行一次。

|  |
| --- |
| $ hexo clean |

### 命令没有执行

如果你除了 hexo help、hexo init 和 hexo version 以外不能执行任何命令、并且你的任何命令都只返回了 hexo help 的内容，这可能是由于 package.json 中缺乏 hexo 字段导致的。

|  |
| --- |
| {  "hexo": {  "version": "3.9.0"  } } |

### 泄露（Escape）内容

Hexo 使用 [Nunjucks](http://mozilla.github.io/nunjucks/" \t "_blank) 来解析文章（旧版本使用 [Swig](http://paularmstrong.github.io/swig/)，两者语法类似），内容若包含 {{ }} 或 {% %} 可能导致解析错误，您可以用 raw 标签包裹来避免潜在问题发生。

|  |
| --- |
| {% raw %} Hello {{ sensitive }} {% endraw %} |

### ENOSPC 错误 （Linux）

运行 $ hexo server 命令有时会返回这样的错误：

|  |
| --- |
| Error: watch ENOSPC ... |

它可以用过运行 $ npm dedupe 来解决，如果不起作用的话，可以尝试在 Linux 终端中运行下列命令：

|  |
| --- |
| $ echo fs.inotify.max\_user\_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p |

这将会提高你能监视的文件数量。

### EMPERM Error (Windows Subsystem for Linux)

在执行 hexo server 后返回如下错误：

|  |
| --- |
| Error: watch /path/to/hexo/theme/ EMPERM |

这是由于你使用的 WSL 版本不支持监听文件系统改动。 最新版的 WSL 已经解决了这一问题。

您也仍然可以通过先使用 hexo g 生成文件然后将其作为静态服务器运行来从 WSL 环境运行服务器：

|  |
| --- |
| $ hexo generate $ hexo server -s |

关于 WSL 的这一 Issue 请前往 <https://github.com/Microsoft/BashOnWindows/issues/216> 查看。目前这一问题已经得到了解决。

### 模板渲染错误

有的时候你在执行 hexo generate 时会返回以下错误信息：

|  |
| --- |
| FATAL Something's wrong. Maybe you can find the solution here: http://hexo.io/docs/troubleshooting.html Template render error: (unknown path) |

这表明你的文件中存在一些不可被识别的字符，比如不可见的零宽度字符。有可能你的新文章存在这个问题，或者你在修改配置文件时导致了这个错误。

检查你的 \_config.yml 文件中是否漏掉了列表前的空格。你可以查阅 Wikipedia 中 [YAML](https://zh.wikipedia.org/wiki/YAML) 相关页面来学习 YAML 语法。

这个是错误的：

|  |
| --- |
| plugins: - hexo-generator-feed - hexo-generator-sitemap |

正确的应该是这样：

|  |
| --- |
| plugins:  - hexo-generator-feed  - hexo-generator-sitemap |

### 开始之前

请遵守以下准则：

* 遵守 [Google JavaScript 代码风格](https://google.github.io/styleguide/jsguide.html)。
* 使用 2 个空格缩排。
* 不要把逗号放在最前面。

### 工作流程

1. Fork [hexojs/hexo](https://github.com/hexojs/hexo" \t "_blank)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

|  |
| --- |
| $ git clone https://github.com/<username>/hexo.git $ cd hexo $ npm install $ git submodule update --init |

1. 新增一个功能分支。

|  |
| --- |
| $ git checkout -b new\_feature |

1. 开始开发。
2. 推送（push）分支。

|  |
| --- |
| $ git push origin new\_feature |

1. 建立一个新的合并申请（pull request）并描述变动。

### 注意事项

* 不要修改 package.json 的版本号。
* 只有在测试通过的情况下您的合并申请才会被批准，在提交前别忘了进行测试。

|  |
| --- |
| $ npm test |

# 设置 RSS

NexT 中 RSS 有三个设置选项，满足特定的使用场景。 更改 主题配置文件，设定 rss 字段的值：

* false：禁用 RSS，不在页面上显示 RSS 连接。
* 留空：使用 Hexo 生成的 Feed 链接。 你可以需要先安装 [hexo-generator-feed](https://github.com/hexojs/hexo-generator-feed) 插件。
* 具体的链接地址：适用于已经烧制过 Feed 的情形。

### 添加「标签」页面

新建「标签」页面，并在菜单中显示「标签」链接。「标签」页面将展示站点的所有标签，若你的所有文章都未包含标签，此页面将是空的。 底下代码是一篇包含标签的文章的例子：

title: 标签测试文章

tags:

- Testing

- Another Tag

---

请参阅 [Hexo 的分类与标签文档](https://hexo.io/zh-cn/docs/front-matter.html#分类和标签)，了解如何为文章添加标签或者分类。

* [新建页面](http://theme-next.iissnan.com/theme-settings.html#new-page-tags)
* [设置页面类型](http://theme-next.iissnan.com/theme-settings.html#set-page-tags)
* [修改菜单](http://theme-next.iissnan.com/theme-settings.html#update-menu-for-tags-page)

在菜单中添加链接。编辑 主题配置文件 ， 添加 tags 到 menu 中，如下:

menu:

home: /

archives: /archives

tags: /tags

**注意：**如果有集成评论服务，页面也会带有评论。 若需要关闭的话，请添加字段 comments 并将值设置为 false，如：

禁用评论示例

title: 标签

date: 2014-12-22 12:39:04

type: "tags"

comments: false

---

### 添加「分类」页面

新建「分类」页面，并在菜单中显示「分类」链接。「分类」页面将展示站点的所有分类，若你的所有文章都未包含分类，此页面将是空的。 底下代码是一篇包含分类的文章的例子：

title: 分类测试文章

categories: Testing

---

请参阅 [Hexo 的分类与标签文档](https://hexo.io/zh-cn/docs/front-matter.html#分类和标签)，了解如何为文章添加标签或者分类。

* [新建页面](http://theme-next.iissnan.com/theme-settings.html#new-page-categories)
* [设置页面类型](http://theme-next.iissnan.com/theme-settings.html#set-page-categories)
* [修改菜单](http://theme-next.iissnan.com/theme-settings.html#update-menu-for-categories-page)

在菜单中添加链接。编辑 主题配置文件 ， 添加 categories 到 menu 中，如下:

menu:

home: /

archives: /archives

categories: /categories

**注意：**如果有集成评论服务，页面也会带有评论。 若需要关闭的话，请添加字段 comments 并将值设置为 false，如：

禁用评论示例

title: 分类

date: 2014-12-22 12:39:04

type: "categories"

comments: false

---

### 设置字体

**注意：** 此特性在版本 5.0.1 中引入，要使用此功能请确保所使用的 NexT 版本在此之后

为了解决 [Google Fonts API](https://www.google.com/fonts) 不稳定的问题，NexT 在 5.0.1 中引入此特性。 通过此特性，你可以指定所使用的字体库外链地址；与此同时，NexT 开放了 5 个特定范围的字体设定，他们是：

* 全局字体：定义的字体将在全站范围使用
* 标题字体：文章内标题的字体（h1, h2, h3, h4, h5, h6）
* 文章字体：文章所使用的字体
* Logo字体：Logo 所使用的字体
* 代码字体： 代码块所使用的字体

各项所指定的字体将作为首选字体，当他们不可用时会自动 Fallback 到 NexT 设定的基础字体组：

* 非代码类字体：Fallback 到 "PingFang SC", "Microsoft YaHei", sans-serif
* 代码类字体： Fallback 到 consolas, Menlo, "PingFang SC", "Microsoft YaHei", monospace

另外，每一项都有一个额外的 external 属性，此属性用来控制是否使用外链字体库。 开放此属性方便你设定那些已经安装在系统中的字体，减少不必要的请求（请求大小）。

配置示例

font:

enable: true

# 外链字体库地址，例如 //fonts.googleapis.com (默认值)

host:

# 全局字体，应用在 body 元素上

global:

external: true

family: Monda

# 标题字体 (h1, h2, h3, h4, h5, h6)

headings:

external: true

family: Roboto Slab

# 文章字体

posts:

external: true

family:

# Logo 字体

logo:

external: true

family: Lobster Two

size: 24

# 代码字体，应用于 code 以及代码块

codes:

external: true

family: PT Mono

### 设置代码高亮主题

NexT 使用 [Tomorrow Theme](https://github.com/chriskempson/tomorrow-theme) 作为代码高亮，共有5款主题供你选择。 NexT 默认使用的是 白色的 normal 主题，可选的值有 normal，night， night blue， night bright， night eighties：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

更改 highlight\_theme 字段，将其值设定成你所喜爱的高亮主题，例如：

高亮主题设置示例

# Code Highlight theme

# Available value: normal | night | night eighties | night blue | night bright

# https://github.com/chriskempson/tomorrow-theme

highlight\_theme: normal

### 侧边栏社交链接

侧栏社交链接的修改包含两个部分，第一是链接，第二是链接图标。 两者配置均在 主题配置文件 中。

1. 链接放置在 social 字段下，一行一个链接。其键值格式是 显示文本: 链接地址。

配置示例

# Social links

social:

GitHub: https://github.com/your-user-name

Twitter: https://twitter.com/your-user-name

微博: http://weibo.com/your-user-name

豆瓣: http://douban.com/people/your-user-name

知乎: http://www.zhihu.com/people/your-user-name

# 等等

1. 设定链接的图标，对应的字段是 social\_icons。其键值格式是 匹配键: Font Awesome 图标名称， 匹配键 与上一步所配置的链接的 显示文本 相同（大小写严格匹配），图标名称 是 Font Awesome 图标的名字（不必带 fa- 前缀）。 enable 选项用于控制是否显示图标，你可以设置成 false 来去掉图标。

配置示例

# Social Icons

social\_icons:

enable: true

# Icon Mappings

GitHub: github

Twitter: twitter

微博: weibo

### 开启打赏功能 由 [habren](https://github.com/iissnan/hexo-theme-next/pull/687) 贡献

越来越多的平台（微信公众平台，新浪微博，简书，百度打赏等）支持打赏功能，付费阅读时代越来越近，特此增加了打赏功能，支持微信打赏和支付宝打赏。 只需要 主题配置文件 中填入 微信 和 支付宝 收款二维码图片地址 即可开启该功能。

打赏功能配置示例

reward\_comment: 坚持原创技术分享，您的支持将鼓励我继续创作！

wechatpay: /path/to/wechat-reward-image

alipay: /path/to/alipay-reward-image

### 友情链接 由 [iamwent](https://github.com/iissnan/hexo-theme-next/pull/250) 贡献

编辑 主题配置文件 添加：

友情链接配置示例

# title

links\_title: Links

links:

MacTalk: http://macshuo.com/

Title: http://example.com/

### 腾讯公益404页面 由 [xirong](https://github.com/iissnan/hexo-theme-next/pull/126) 贡献

腾讯公益404页面，寻找丢失儿童，让大家一起关注此项公益事业！效果如下 <http://www.ixirong.com/404.html>

使用方法，新建 404.html 页面，放到主题的 source 目录下，内容如下：

<!DOCTYPE HTML>

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=utf-8;"/>

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<meta name="robots" content="all" />

<meta name="robots" content="index,follow"/>

<link rel="stylesheet" type="text/css" href="https://qzone.qq.com/gy/404/style/404style.css">

</head>

<body>

<script type="text/plain" src="http://www.qq.com/404/search\_children.js"

charset="utf-8" homePageUrl="/"

homePageName="回到我的主页">

</script>

<script src="https://qzone.qq.com/gy/404/data.js" charset="utf-8"></script>

<script src="https://qzone.qq.com/gy/404/page.js" charset="utf-8"></script>

</body>

</html>

### 站点建立时间

这个时间将在站点的底部显示，例如 © 2013 - 2015。 编辑 主题配置文件，新增字段 since。

配置示例

since: 2013

### 订阅微信公众号 由 [huiwang](https://github.com/iissnan/hexo-theme-next/pull/788) 贡献

**注意：** 此特性在版本 5.0.1 中引入，要使用此功能请确保所使用的 NexT 版本在此之后

在每篇文章的末尾显示微信公众号二维码，扫一扫，轻松订阅博客。

在微信公众号平台下载您的二维码，并将它存放于博客source/uploads/目录下。

然后编辑 主题配置文件，如下：

配置示例

# Wechat Subscriber

wechat\_subscriber:

enabled: true

qcode: /uploads/wechat-qcode.jpg

description: 欢迎您扫一扫上面的微信公众号，订阅我的博客！

### 设置「动画效果」

NexT 默认开启动画效果，效果使用 JavaScript 编写，因此需要等待 JavaScript 脚本完全加载完毕后才会显示内容。 如果您比较在乎速度，可以将设置此字段的值为 false 来关闭动画。

编辑 主题配置文件， 搜索 use\_motion，根据您的需求设置值为 true 或者 false 即可：

use\_motion: true # 开启动画效果

use\_motion: false # 关闭动画效果

### 设置「背景动画」

NexT 自带两种背景动画效果

编辑 主题配置文件， 搜索 canvas\_nest 或 three\_waves，根据您的需求设置值为 true 或者 false 即可：

**注意：** three\_waves 在版本 5.1.1 中引入。只能同时开启一种背景动画效果。

canvas\_nest 配置示例

# canvas\_nest

canvas\_nest: true //开启动画

canvas\_nest: false //关闭动画

three\_waves 配置示例

# three\_waves

three\_waves: true //开启动画

three\_waves: false //关闭动画

# Hexo指令

## init

|  |
| --- |
| $ hexo init [folder] |

新建一个网站。如果没有设置 folder ，Hexo 默认在目前的文件夹建立网站。

## new

|  |
| --- |
| $ hexo new [layout] <title> |

新建一篇文章。如果没有设置 layout 的话，默认使用 [\_config.yml](https://hexo.io/zh-cn/docs/configuration) 中的 default\_layout 参数代替。如果标题包含空格的话，请使用引号括起来。

|  |
| --- |
| $ hexo new "post title with whitespace" |

| **参数** | **描述** |
| --- | --- |
| -p, --path | 自定义新文章的路径 |
| -r, --replace | 如果存在同名文章，将其替换 |
| -s, --slug | 文章的 Slug，作为新文章的文件名和发布后的 URL |

默认情况下，Hexo 会使用文章的标题来决定文章文件的路径。对于独立页面来说，Hexo 会创建一个以标题为名字的目录，并在目录中放置一个 index.md 文件。你可以使用 --path 参数来覆盖上述行为、自行决定文件的目录：

|  |
| --- |
| hexo new page --path about/me "About me" |

以上命令会创建一个 source/about/me.md 文件，同时 Front Matter 中的 title 为 "About me"

注意！title 是必须指定的！如果你这么做并不能达到你的目的：

|  |
| --- |
| hexo new page --path about/me |

此时 Hexo 会创建 source/\_posts/about/me.md，同时 me.md 的 Front Matter 中的 title 为 "page"。这是因为在上述命令中，hexo-cli 将 page 视为指定文章的标题、并采用默认的 layout。

## generate

|  |
| --- |
| $ hexo generate |

生成静态文件。

| **选项** | **描述** |
| --- | --- |
| -d, --deploy | 文件生成后立即部署网站 |
| -w, --watch | 监视文件变动 |
| -b, --bail | 生成过程中如果发生任何未处理的异常则抛出异常 |
| -f, --force | 强制重新生成文件 Hexo 引入了差分机制，如果 public 目录存在，那么 hexo g 只会重新生成改动的文件。 使用该参数的效果接近 hexo clean && hexo generate |
| -c, --concurrency | 最大同时生成文件的数量，默认无限制 |

该命令可以简写为

|  |
| --- |
| $ hexo g |

## publish

|  |
| --- |
| $ hexo publish [layout] <filename> |

发表草稿。

## server

|  |
| --- |
| $ hexo server |

启动服务器。默认情况下，访问网址为： http://localhost:4000/。

| **选项** | **描述** |
| --- | --- |
| -p, --port | 重设端口 |
| -s, --static | 只使用静态文件 |
| -l, --log | 启动日记记录，使用覆盖记录格式 |

## deploy

|  |
| --- |
| $ hexo deploy |

部署网站。

| **参数** | **描述** |
| --- | --- |
| -g, --generate | 部署之前预先生成静态文件 |

该命令可以简写为：

|  |
| --- |
| $ hexo d |

## render

|  |
| --- |
| $ hexo render <file1> [file2] ... |

渲染文件。

| **参数** | **描述** |
| --- | --- |
| -o, --output | 设置输出路径 |

## migrate

|  |
| --- |
| $ hexo migrate <type> |

从其他博客系统 [迁移内容](https://hexo.io/zh-cn/docs/migration)。

## clean

|  |
| --- |
| $ hexo clean |

清除缓存文件 (db.json) 和已生成的静态文件 (public)。

在某些情况（尤其是更换主题后），如果发现您对站点的更改无论如何也不生效，您可能需要运行该命令。

## list

|  |
| --- |
| $ hexo list <type> |

列出网站资料。

## version

|  |
| --- |
| $ hexo version |

显示 Hexo 版本。

## 选项

### 安全模式

|  |
| --- |
| $ hexo --safe |

在安全模式下，不会载入插件和脚本。当您在安装新插件遭遇问题时，可以尝试以安全模式重新执行。

### 调试模式

|  |
| --- |
| $ hexo --debug |

在终端中显示调试信息并记录到 debug.log。当您碰到问题时，可以尝试用调试模式重新执行一次，并 [提交调试信息到 GitHub](https://github.com/hexojs/hexo/issues/new)。

### 简洁模式

|  |
| --- |
| $ hexo --silent |

隐藏终端信息。

### 自定义配置文件的路径

|  |
| --- |
| # 使用 custom.yml 代替默认的 \_config.yml $ hexo server --config custom.yml  # 使用 custom.yml 和 custom2.json，其中 custom2.json 优先级更高 $ hexo generate --config custom.yml,custom2.json,custom3.yml |

自定义配置文件的路径，指定这个参数后将不再使用默认的 \_config.yml。  
你可以使用一个 YAML 或 JSON 文件的路径，也可以使用逗号分隔（无空格）的多个 YAML 或 JSON 文件的路径。例如：

|  |
| --- |
| # 使用 custom.yml 代替默认的 \_config.yml $ hexo server --config custom.yml  # 使用 custom.yml, custom2.json 和 custom3.yml，其中 custom3.yml 优先级最高，其次是 custom2.json $ hexo generate --config custom.yml,custom2.json,custom3.yml |

当你指定了多个配置文件以后，Hexo 会按顺序将这部分配置文件合并成一个 \_multiconfig.yml。如果遇到重复的配置，排在后面的文件的配置会覆盖排在前面的文件的配置。这个原则适用于任意数量、任意深度的 YAML 和 JSON 文件。

### 显示草稿

|  |
| --- |
| $ hexo --draft |

显示 source/\_drafts 文件夹中的草稿文章。

### 自定义 CWD

|  |
| --- |
| $ hexo --cwd /path/to/cwd |

自定义当前工作目录（Current working directory）的路径。

# 第三方服务

## 评论系统

NexT 支持多款评论系统。

如需取消某个 页面/文章 的评论，在 md 文件的 [front-matter](https://hexo.io/docs/front-matter.html) 中增加 comments: false

### DISQUS

* [>= 5.1.1](http://theme-next.iissnan.com/third-party-services.html#disqus-new-settings)
* [< 5.1.1](http://theme-next.iissnan.com/third-party-services.html#disqus-old-settings)

编辑 主题配置文件， 将 disqus 下的 enable 设定为 true，同时提供您的 shortname。count 用于指定是否显示评论数量。

disqus:

enable: false

shortname:

count: true

### Facebook Comments 由 [hydai](https://github.com/iissnan/hexo-theme-next/pull/925) 贡献

需要先集成 [Facebook SDK](http://theme-next.iissnan.com/third-party-services.html#facebook-sdk)。修改 主题配置文件 以下配置：

# Facebook comments plugin

# This plugin depends on Facebook SDK.

# If facebook\_sdk.enable is false, Facebook comments plugin is unavailable.

facebook\_comments\_plugin:

enable: false

num\_of\_posts: 10 # min posts num is 1

width: 100% # default width is 550px

scheme: light # default scheme is light (light or dark)

### HyperComments 由 [ivan-nginx](https://github.com/iissnan/hexo-theme-next/pull/1155) 贡献

Add support for [Hypercomments](https://www.hypercomments.com/) system instead of Disqus. 修改 主题配置文件 以下配置：

# Hypercomments

#hypercomments\_id:

### 网易云跟帖 由 [geekrainy](https://github.com/iissnan/hexo-theme-next/pull/1301) 贡献

登陆 [网易云跟帖](https://gentie.163.com/) 获取你的 Product Key。 编辑 主题配置文件， 编辑 gentie\_productKey 字段，设置如下：

gentie\_productKey: #your-gentie-product-key

请注意，您在云跟帖管理后台设置的域名必须跟您站点的域名一致。在本地测试时，需要做两步骤前置设定：

1. 修改 hosts 文件，将您域名的请求指向本地。例如：127.0.0.1 notes.example.com
2. 修改 Hexo 监听的端口为 80：hexo s --debug -p 80

测试完成后请将 hosts 文件中的域名映射删除。

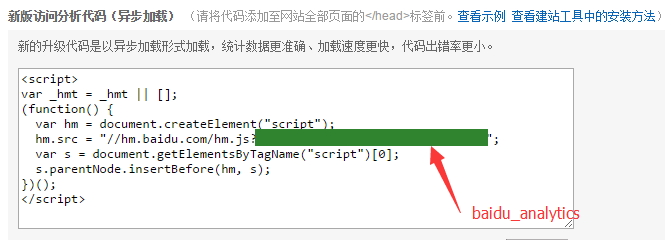
### 来必力 由 [asmoker](https://github.com/iissnan/hexo-theme-next/pull/1415) 贡献

登陆 [来必力](https://livere.com/) 获取你的 LiveRe UID。 编辑 主题配置文件， 编辑 livere\_uid 字段，设置如下：

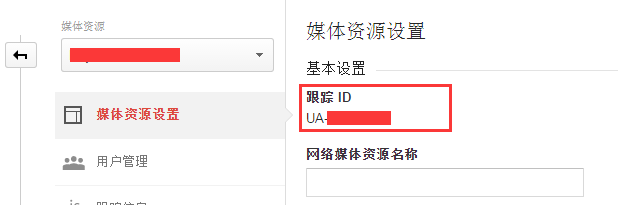
livere\_uid: #your livere\_uid

## 数据统计与分析

### 百度统计

1. 登录 [百度统计](http://tongji.baidu.com/)，定位到站点的代码获取页面
2. 复制 hm.js? 后面那串统计脚本 id，如下图所示：  
   
3. 编辑 主题配置文件， 修改字段 baidu\_analytics，值设置成你的百度统计脚本 id。

### Google 分析

编辑 主题配置文件， 修改字段 google\_analytics， 值设置成你的 Google 跟踪 ID。跟踪 ID 通常是以 UA- 开头。 

### 腾讯分析 由 [Cissoid](https://github.com/iissnan/hexo-theme-next/pull/603) 贡献

请登录 [腾讯分析](http://ta.qq.com/)，登录并获取分析的 ID。 然后在 主题配置文件 里将 ID 放置 tencent\_analytics 字段。

### CNZZ 统计 由 [jerrybendy](https://github.com/iissnan/hexo-theme-next/pull/712) 贡献

在 主题配置文件 中增加了一项 cnzz\_siteid的配置项，值为 CNZZ 里面添加统计的站点ID。 这个ID可以在地址栏里，或者自动生成的脚本里面找到。

默认脚本生成的内容会产生“站长统计”几个字，感觉很丑而且和主题不太搭，我用了一个 display: none;把这几个字给隐藏了。

### 不蒜子统计 由 [panzhitian](https://github.com/iissnan/hexo-theme-next/pull/809) 贡献

**注意：** 此特性在版本 5.0.1 中引入，要使用此功能请确保所使用的 NexT 版本在此之后

* [全局配置](http://theme-next.iissnan.com/third-party-services.html#analytics-busuanzi-enable)
* [站点UV配置](http://theme-next.iissnan.com/third-party-services.html#analytics-busuanzi-site-uv)
* [站点PV配置](http://theme-next.iissnan.com/third-party-services.html#analytics-busuanzi-site-pv)
* [单页面PV配置](http://theme-next.iissnan.com/third-party-services.html#analytics-busuanzi-page-pv)

编辑 主题配置文件 中的busuanzi\_count的配置项。

当enable: true时，代表开启全局开关。若site\_uv、site\_pv、page\_pv的值均为false时，不蒜子仅作记录而不会在页面上显示。

### 腾讯移动分析 由 [aleonchen](https://github.com/iissnan/hexo-theme-next/pull/1408) 贡献

编辑 主题配置文件，填写 tencent\_mta 的值。

# Tencent MTA ID

tencent\_mta: your-tencent-mta-id

### 阅读次数统计（LeanCloud) 由 [Doublemine](https://github.com/iissnan/hexo-theme-next/pull/439) 贡献

请查看 [为NexT主题添加文章阅读量统计功能](https://notes.wanghao.work/2015-10-21-%E4%B8%BANexT%E4%B8%BB%E9%A2%98%E6%B7%BB%E5%8A%A0%E6%96%87%E7%AB%A0%E9%98%85%E8%AF%BB%E9%87%8F%E7%BB%9F%E8%AE%A1%E5%8A%9F%E8%83%BD.html#%E9%85%8D%E7%BD%AELeanCloud)

## 内容分享服务

### JiaThis

编辑 主题配置文件， 添加/修改字段 jiathis，值为 true。

JiaThis 内容分享服务配置示例

# JiaThis 分享服务

jiathis: true

### [百度分享](http://share.baidu.com/)

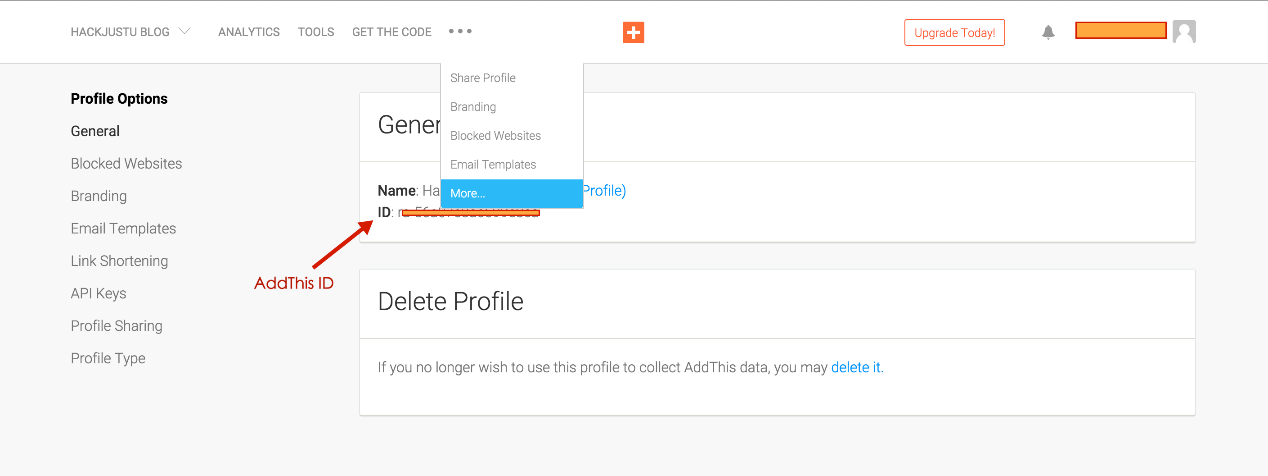
编辑 主题配置文件，添加/修改字段 baidushare，值为 true。

百度内容分享服务配置示例

# 百度分享服务

baidushare: true

### AddThis 由 [hackjustu](https://github.com/iissnan/hexo-theme-next/pull/660) 贡献

1. 在网站 [AddThis](https://www.addthis.com/) 上注册账号。 可以使用 Google/Facebook/Twitter 账号进行第三方登陆
2. 从下面菜单获得 AddThis id：More.. --> General --> ID。 具体 ID 获得方式参考以下截图   
   
3. 在 主题配置文件 中，把#Share下的 #add\_this\_id取消注释， 改为add\_this\_id: put\_your\_add\_this\_id\_here

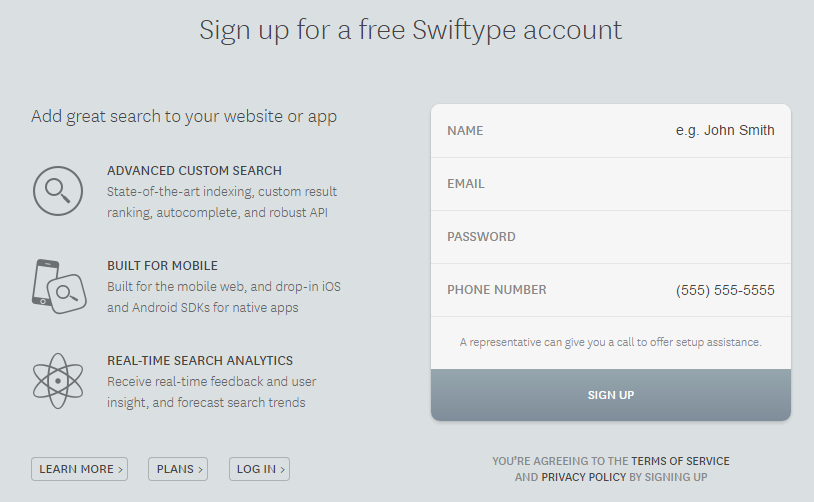
## 搜索服务

NexT 支持集成 [Swiftype](https://swiftype.com/)、 [微搜索](http://tinysou.com/)、Local Search 和 Algolia。

### Swiftype 站内搜索 由 [lzlun129](https://github.com/iissnan/hexo-theme-next/pull/84) 贡献

* [注册账户](http://theme-next.iissnan.com/third-party-services.html#swiftype-step-1)
* [创建搜索引擎](http://theme-next.iissnan.com/third-party-services.html#swiftype-step-2)
* [定制与激活搜索](http://theme-next.iissnan.com/third-party-services.html#swiftype-step-3)
* [获取 Key](http://theme-next.iissnan.com/third-party-services.html#swiftype-step-4)
* [更新主题配置](http://theme-next.iissnan.com/third-party-services.html#swiftype-step-5)

前往 [Swiftype 注册页面](https://swiftype.com/users/sign_up)，注册一个新账户。



### 微搜索 由 [lzlun129](https://github.com/iissnan/hexo-theme-next/pull/84) 贡献

TBD

### Local Search 由 [flashlab](https://github.com/iissnan/hexo-theme-next/pull/694) 贡献

添加百度/谷歌/本地 自定义站点内容搜索

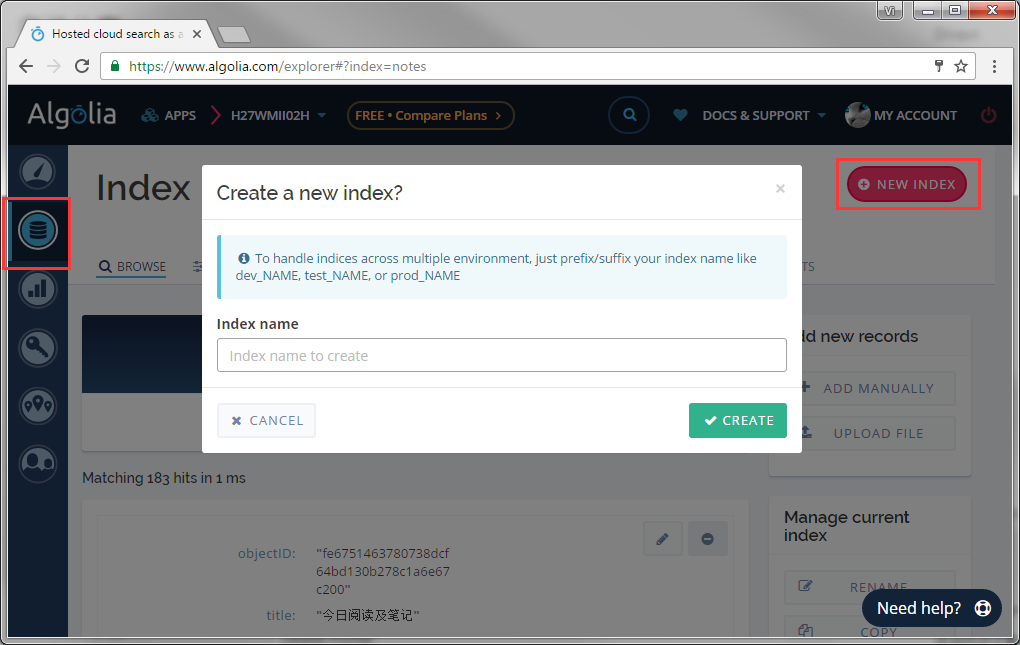
1. 安装 hexo-generator-searchdb，在站点的根目录下执行以下命令：
2. $ npm install hexo-generator-searchdb --save
3. 编辑 站点配置文件，新增以下内容到任意位置：
4. search:
5. path: search.xml
6. field: post
7. format: html
8. limit: 10000
9. 编辑 主题配置文件，启用本地搜索功能：
10. # Local search
11. local\_search:
12. enable: true

### Algolia

**注意：** 此特性在版本 5.1.0 中引入，要使用此功能请确保所使用的 NexT 版本在此之后

* [注册 Algolia，创建 Index](http://theme-next.iissnan.com/third-party-services.html#algolia-step-1)
* [安装 Hexo Algolia](http://theme-next.iissnan.com/third-party-services.html#algolia-step-2)
* [获取 Key，更新站点配置](http://theme-next.iissnan.com/third-party-services.html#algolia-step-3)
* [更新 Index](http://theme-next.iissnan.com/third-party-services.html#algolia-step-4)
* [主题集成](http://theme-next.iissnan.com/third-party-services.html#algolia-step-5)

前往 [Algolia 注册页面](https://www.algolia.com)，注册一个新账户。 可以使用 GitHub 或者 Google 账户直接登录，注册后的 14 天内拥有所有功能（包括收费类别的）。之后若未续费会自动降级为免费账户，免费账户 总共有 10,000 条记录，每月有 100,000 的可以操作数。注册完成后，创建一个新的 Index，这个 Index 将在后面使用。



## 其他服务

### MathJax

NexT 借助于 MathJax 来显示数学公式，此选项默认关闭。

* [5.0.0 及之前的版本](http://theme-next.iissnan.com/third-party-services.html#mathjax-old-settings)
* [5.0.1 及以上的版本](http://theme-next.iissnan.com/third-party-services.html#mathjax-new-settings)

编辑 主题配置文件，将 mathjax 设定为 true 即可。

# MathJax Support

mathjax:

ProTip: [使用七牛 CDN 来加速 MathJax 脚本的加载](https://github.com/iissnan/hexo-theme-next/pull/363)

### Facebook SDK 由 [changyuheng](https://github.com/iissnan/hexo-theme-next/pull/410) 贡献

修改 主题配置文件 以下配置：

# Facebook SDK Support.

facebook\_sdk:

enable: false

app\_id:

fb\_admin:

like\_button: #true

webmaster: #true

### Google Webmaster tools

设置 [Google站点管理工具](https://www.google.com/webmasters/tools/) 的验证字符串，用于提交 sitemap

1. 获取 google site verification code  
   登录 Google Webmaster Tools，导航到验证方法，并选择 HTML Tag。将会获取到一段代码：

<meta name="google-site-verification" content="XXXXXXXXXXXXXXXXXXXXXXX" />

将 content 里面的 XXXXXXXXXXXXXXXXXXXXXXX 复制出来。

1. 设置主题。编辑 主题配置文件，新增/修改字段 google\_site\_verification。
2. google\_site\_verification: XXXXXXXXXXXXXXXXXXXXXXX

# 设置 「JavaScript 第三方库」

**注意：** 此特性在版本 5.0.1 中引入，要使用此功能请确保所使用的 NexT 版本在此之后

设定成合适的 CDN 地址，此特性可以加速静态资源（JavaScript 第三方库）的加载，目前 NexT 使用的第三方库包括：

| **名称** | **官方地址** | **NexT 内部使用版本** |
| --- | --- | --- |
| jQuery | <http://jquery.com/> | 2.1.3 |
| Fancybox | <http://fancyapps.com/fancybox/> | 2.1.5 |
| FastClick | <https://github.com/ftlabs/fastclick> | 1.0.6 |
| jQuery LazyLoad | <https://github.com/tuupola/jquery_lazyload> | 2.1.5 |
| VelocityJS | <http://julian.com/research/velocity/> | 1.2.1 |
| VelocityJS UI Pack | <http://julian.com/research/velocity/#uiPack> | 1.2.1 |
| UAParser.js | <https://faisalman.github.io/ua-parser-js/> | 0.7.9 |
| FontAwesome | <http://fontawesome.io/> | 4.4.0 |

此特性配置位于 主题配置文件，键值 vendors。 其中 \_internal: vendors 用于指定 NexT 内部使用的路径前缀，通常情况下不需要更改。接着便是各个库的设定，若设定为空，NexT 将使用内部集成的文件。每一个设定都是 库： CDN 地址 的键值形式。例如：

jquery: //cdn.jsdelivr.net/jquery/2.1.3/jquery.min.js

「Internal version」说明 NexT 内部使用的版本，经过测试的版本。因此若你需要指定其他版本，需要进行额外的自行测试。

设置示例

vendors:

# Internal path prefix. Please do not edit it.

\_internal: vendors

# Internal version: 2.1.3

jquery: //cdn.jsdelivr.net/jquery/2.1.3/jquery.min.js

# Internal version: 2.1.5

# Fancybox: http://fancyapps.com/fancybox/

fancybox: //cdn.jsdelivr.net/fancybox/2.1.5/jquery.fancybox.pack.js

fancybox\_css: //cdn.jsdelivr.net/fancybox/2.1.5/jquery.fancybox.min.css

# Internal version: 1.0.6

fastclick: //cdn.jsdelivr.net/fastclick/1.0.6/fastclick.min.js

# Internal version: 1.9.7

lazyload: //cdn.jsdelivr.net/jquery.lazyload/1.9.3/jquery.lazyload.min.js

# Internal version: 1.2.1

velocity: //cdn.jsdelivr.net/velocity/1.2.3/velocity.min.js

# Internal version: 1.2.1

velocity\_ui: //cdn.jsdelivr.net/velocity/1.2.3/velocity.ui.min.js

# Internal version: 0.7.9

ua\_parser: //cdn.jsdelivr.net/ua-parser.js/0.7.10/ua-parser.min.js

# Internal version: 4.4.0

# http://fontawesome.io/

fontawesome: //maxcdn.bootstrapcdn.com/font-awesome/4.6.2/css/font-awesome.min.css

# Internal version: 1

# https://www.algolia.com

algolia\_instant\_js:

algolia\_instant\_css:

# Internal version: 1.0.0

# https://github.com/hustcc/canvas-nest.js

canvas\_nest:

# Internal version: r84

# https://github.com/mrdoob/three.js/releases

three:

# Internal version: 1.0.0

# https://github.com/jjandxa/three\_waves

three\_waves:

## 如何设置「阅读全文」？

在首页显示一篇文章的部分内容，并提供一个链接跳转到全文页面是一个常见的需求。 NexT 提供三种方式来控制文章在首页的显示方式。 也就是说，在首页显示文章的摘录并显示 **阅读全文** 按钮，可以通过以下方法：

1. 在文章中使用 <!-- more --> 手动进行截断，Hexo 提供的方式 推荐
2. 在文章的 [front-matter](https://hexo.io/docs/front-matter.html) 中添加 description，并提供文章摘录
3. 自动形成摘要，在 主题配置文件 中添加：
4. auto\_excerpt:
5. enable: true
6. length: 150

默认截取的长度为 150 字符，可以根据需要自行设定

建议使用 <!-- more -->（即第一种方式），除了可以精确控制需要显示的摘录内容以外， 这种方式也可以让 Hexo 中的插件更好的识别。

## Favicon 设置后没有生效？

将你的 favicon 放置到 **站点** 的 source 目录下，与站点的配置文件同级。 若你发现设置的 Favicon 并未生效，请先清除浏览器的缓存后直接访问 Favicon 的地址，这个地址通常是 http://your-domain.com/favicon.ico。

如果你的站点是放在子目录下，请设置为：favicon: favicon.ico

## 如何更改字体？

NexT 从 **5.0.1** 版本开始提供一个 [字体定制特性](http://theme-next.iissnan.com/theme-settings.html#fonts-customization)， 请先查看此特性是否能满足你的需求。以下的修改将覆盖 字体定制 的特性。 编辑主题下的 source/css/\_variables/custom.styl 文件，新增两个变量：

// 标题，修改成你期望的字体族

$font-family-headings = Georgia, sans

// 修改成你期望的字体族

$font-family-base = "Microsoft YaHei", Verdana, sans-serif

// 代码字体

$code-font-family = "Input Mono", "PT Mono", Consolas, Monaco, Menlo, monospace

// 正文字体的大小

$font-size-base = 16px

// 代码字体的大小

$code-font-size = 13px

## 如何更改内容区域的宽度？

NexT 对于内容的宽度的设定如下：

* 700px，当屏幕宽度 < 1600px
* 900px，当屏幕宽度 >= 1600px
* 移动设备下，宽度自适应

如果你需要修改内容的宽度，同样需要编辑样式文件。 编辑主题的 source/css/\_variables/custom.styl 文件，新增变量：

// 修改成你期望的宽度

$content-desktop = 700px

// 当视窗超过 1600px 后的宽度

$content-desktop-large = 900px

此方法不适用于 Pisces Scheme，关于如何修改 Pisces Scheme 的宽度请参看 [这个 Issue](https://github.com/iissnan/hexo-theme-next/issues/759#issuecomment-202242848)

## 为何新增的菜单项显示 Menu.xxx

NexT 使用了 Hexo 提供的国际化功能，因此在 Menu 部分配置的值并非直接用于界面的展示，而是作为一个名称。 当渲染的时候，NexT 去寻找对应的语言文件，并替换这个键所定义的文本后展示。

以 home 为例，在 主题配置文件 中配置为 home: /， 此处的键即是 home，在不同的语言中，这个键所对应的文本都不同：

# 简体中文

# 语言文件 languages/zh-Hans.yml

menu:

home: 首页

# 英文

# 语言文件 languages/en.yml

menu:

home: Home

因此，当新增一个项时，必须在 **所使用的语言** 中添加对应的翻译文本。

关于菜单的配置，请参考 [菜单的配置](http://theme-next.iissnan.com/getting-started.html#menu-settings)

## 标签/分类数量统计不准确？

因为 Hexo 有缓存的功能，因此有时候你会发现在 标签 和 分类 页面中的数量统计并不准确。 出现这个问题时，可以按照以下步骤重新生成站点的内容：

1. 删除站点目录下的 db.json 文件
2. 在站点目录下执行命令 hexo clean
3. 在站点目录下执行命令，重新生成 hexo generate

当执行完以上步骤后，可以在本地启动服务器来验证下是否已经解决问题。

## 为何语言显示不正确？

若没有设定 language 的值或者 language: default，会出现语言选择不正确的情况。 因此为了确保语言正确显示，请明确指定需要使用的语言。 编辑 站点配置文件， 将 language 字段设置成明确的语言代码。 例如，设置语言为简体中文：

language: zh-Hans

| **语言** | **对应值** |
| --- | --- |
| English | en |
| 简体中文 | zh-Hans |
| 法语 | fr-FR |
| 繁体中文 | zh-hk/zh-tw |
| 俄语 | ru |
| 德语 | de |
| 葡萄牙语 | pt |
| 日语 | ja |

## 如何关闭新建页面的评论功能？

当集成了评论系统，如 Disqus，所有新建的页面都将自动开启评论。若你不需要评论，请在页面的 Front-matter 里添加 comments 字段，并将值设置为 false。如下所示：

title: All tags

date: 2015-12-16 17:05:24

type: "tags"

comments: false

---

## 如何设置页面文章的篇数？

在 Hexo 里可以为首页和归档页面设置不同的文章篇数，但可能需要安装 Hexo 插件。详细步骤如下。

1. 使用 npm install --save 命令来安装需要的 Hexo 插件。
2. npm install --save hexo-generator-index
3. npm install --save hexo-generator-archive
4. npm install --save hexo-generator-tag
5. 等待扩展全部安装完成后，在 **站点配置文章** 中，设定如下选项：
6. index\_generator:
7. per\_page: 5
8. archive\_generator:
9. per\_page: 20
10. yearly: true
11. monthly: true
12. tag\_generator:
13. per\_page: 10

per\_page 即文章的数量。

需要注意的是，Hexo 3.2 版本不允许配置文件中存在重复的选项设置。 因此，最好检查下 站点配置文件 中是否有存在上述同名的配置。 如果存在，请将两者配置在一起。

## 如何优化 NexT 主题？

NexT 并未对 HTML、JavaScript 以及 CSS 做压缩处理。如果你比较在意这点，可以在 NexT 的 Issues 列表搜索 [他人共享的优化方案](https://github.com/iissnan/hexo-theme-next/issues?utf8=%E2%9C%93&q=+label%3AOptimization) 。

## Cannot find module XXX

NexT 提供的标签插件引用了 Hexo 模块中的一些子模块，例如 hexo-util。 NPM 在版本 3 中引入了新的模块安装机制， NexT 将使用这种机制，意味着如果是旧的 NPM 将会碰到这个错误。 因此，如果您碰到 "Cannot find module XXX" 的错误时，请首先使用 npm -v 确认您的 NPM 版本。

* 版本 > 3 依然报错： 请先删除 node\_module 目录，然后使用 npm install 重新安装一下模块。
* 版本 < 3： 您可以选择升级您的 NPM； 或者在站点目录下明确指定模块依赖 npm install --save hexo-util。 其中 `hexo-util` 仅是示例，请替换成错误中提示的模块名称。

# 更新文档

Hexo 文档开放源代码，您可以在 [hexojs/site](https://github.com/hexojs/site" \t "_blank) 找到源代码。

## 工作流程

1. Fork [hexojs/site](https://github.com/hexojs/site" \t "_blank)
2. 把库（repository）复制到电脑上，并安装所依赖的插件。

|  |
| --- |
| $ git clone https://github.com/<username>/site.git $ cd site $ npm install |

1. 开始编辑文件，您可以通过服务器预览变动。

|  |
| --- |
| $ hexo server |

1. 推送（push）分支。
2. 建立一个新的合并申请（pull request）并描述变动。

## 翻译

1. 在 source 资料夹中建立一个新的语言资料夹（全小写）。
2. 把 source 资料夹中相关的文件（Markdown 和模板文件）复制到新的语言资料夹中。
3. 在 source/\_data/language.yml 中新增语言。
4. 将 en.yml 复制到 themes/navy/languages中并命名为语言名称（全小写）。

# 反馈问题

当您在使用 Hexo 时遇到问题，您可以尝试在 [问题解答](https://hexo.io/zh-cn/docs/troubleshooting) 中寻找解答，或是在 [GitHub](https://github.com/hexojs/hexo/issues) 或 [Google Group](https://groups.google.com/group/hexo) 上提问。如果你没有找答案，请在 Github 报告它。

1. 在 [调试模式](https://hexo.io/zh-cn/docs/commands#%E8%B0%83%E8%AF%95%E6%A8%A1%E5%BC%8F) 中重现问题。
2. 运行 hexo version 并检查版本信息。
3. 把调试信息和版本信息都贴到 GitHub。

# Hexo的数据DB扩展查询

我们已经知道，Hexo已经为我们预先定义了很多常用的变量供我们使用，具体可以在 Hexo | 变量 查询。但是如果系统提供的变量数据不能满足我们的要求，那我们该怎么办呢？其实我们可以通过扩展查询来获取到我们期望的数据。

其实Hexo所有的文章分类标签等等变量信息，在编译成本地静态文件之前，都是本地存储在一个db.json中的，相当于小型的本地数据库，Hexo在运行阶段，所有的数据相关操作其实都是在这个小型数据库上进行操作，其底层使用的查询引擎就是Warehouse。因此我们可以通过Warehouse的语法进行自定义扩展查询。

比如我们需要在页面的底部展示全站的最近6篇文章列表，由于Hexo首页只提供了第一页的数据，因此我们可以基于site变量进行扩展查询：

site.posts.sort({date: -1}).limit(6)

site.posts表示所有的文章，sort（{date: -1})表示按创建时间倒序排列，limit(6)表示只取前6条数据，这样我们就可以拿到了全站的最近6文章信息，后续进行相应展示操作即可。

其他更多复杂的扩展查询都可以根据Warehouse语法文档进行按需扩展。

# 新建仓库

都说了要给他人访问，如果你连个仓库都没有，是不是说不过去？出门右转上 github 新建一个仓库，仓库名必须为 <user-name>.github.io 格式，其中 <user-name> 是你 github 的昵称。不要问我为什么，问我我也不想回答你。

## 2. 全局安装hexo

如果你连 npm 是啥都不知道，对不起，我们的缘分尽了，你走吧。...别，别，别，开个玩笑，有啥问题可以留言。打开命令行，输入下面的命令，全局安装 hexo。

npm install -g hexo

## 3. 初始化项目

Just one 命令，hexo init，就会自动构建一个 hexo 项目，紧接着执行 hexo s，浏览器访问 localhost:4000 就可以看到效果啦！过程一定要快，凉了就不好吃了！

hexo init

// 本地运行

hexo s

## 4. 部署到github

虽然只有一份 Hello World，但是我也想让全世界的人看到，成为最闪耀的那个崽，可咋整？不要慌，快速在项目根目录下找到 \_congif.yml，找到 deploy 字段并填写完整，下面请开始你上学时最擅长的事情：抄写！

# Deployment

## Docs: https://hexo.io/docs/deployment.html

deploy:

type: git

repo: <你的仓库地址> # https://github.com/TJ-XiaJiaHao/TJ-XiaJiaHao.github.io

branch: master

抄写完毕！但是！我们需要额外的一个工具来帮助我们推到仓库上，那就是！那就是！那就是 hexo-deployer-git。搞它！

npm install hexo-deployer-git --save

万事俱备，只欠东南风！执行下面两个命令，就可以把项目自动部署到 github 上啦啦啦啦啦。

hexo clean

hexo deploy

## 5. 查看效果

凡是连环杀手都喜欢在杀人后留在现场，因为他喜欢欣赏自己的杰作！

浏览器访问：https://tj-xiajiahao.github.io/ 即可看到效果。

至此，已经完成了个人博客的搭建，也可以在浏览器中访问，下面将介绍一些常用的野外生存技能。

## 如何创建新文章

阳光明媚，一杯温暖的咖啡，一束倾斜的阳光，惬意地抿一口咖啡，我要开始写博客了！

// [layout] 为布局，可选项为 `post`、`page`、`draft`，这将决定文章所在文件路径。

// <title> 为文章标题

// 如 hexo new post 除了帅气，我还有啥！

hexo new [layout] <title>

## 更换主题皮肤

风格不喜欢？换之。

更换主题流程：下载主题 -> 配置主题，以 xoxo 为例（不要想歪了！）

# 下载到themes文件夹下

git clone https://github.com/KevinOfNeu/hexo-theme-xoxo xoxo

# 修改 \_config.yml 配置

theme: xoxo

## 部署优化

每次都要执行 hexo clean 和 hexo deploy，不如写个新的脚本

// package.json

"dev": "hexo s",

"build": "hexo clean & hexo deploy"

部署命令

npm run build

## 接入评论系统 - valine

<https://valine.js.org/quickstart.html>

# 将Hexo部署到个人网站上

首先安装 [hexo-deployer-sftp](https://link.juejin.im/?target=https%3A%2F%2Fgithub.com%2Flucascaro%2Fhexo-deployer-sftp)

1. npm install hexo-deployer-sftp *--save*
2. 复制代码

## 修改配置

(**记得冒号后必须有空格，不然配置不生效，这是Hexo的一个坑！**）

1. deploy:
2. type: sftp
3. host: <host>
4. user: <user>
5. pass: <password>
6. remotePath: [remote path]
7. port: [port]
8. privateKey: [path/to/privateKey]
9. passphrase: [passphrase]
10. agent: [path/to/agent/socket]
11. 复制代码

host 远程主机的地址

user 使用者名称

pass 密码

remotePath 远程主机的根目录

port 端口

privateKey ssh私钥的目录地址

passphrase （可省略）ssh私钥的密码短语

agent ssh套接字的目录地址

## 执行部署命令

输入hexo g -d就可以生成并部署到自己的个人服务器上了

# Hexo资源

https://hexo.io/zh-cn/docs/configuration.html

# Hexo主题资源

按评价排序

## 主题NexT

http://theme-next.iissnan.com/getting-started.html

[Hexo](https://hexo.io/zh-cn/) 是高效的静态站点生成框架，她基于 [Node.js](https://nodejs.org)。 通过 Hexo 你可以轻松地使用 Markdown 编写文章，除了 Markdown 本身的语法之外，还可以使用 Hexo 提供的 [标签插件](https://hexo.io/zh-cn/docs/tag-plugins.html) 来快速的插入特定形式的内容。在这篇文章中，假定你已经成功安装了 Hexo，并使用 Hexo 提供的命令创建了一个站点。

你可以访问 [Hexo 的文档](https://hexo.io/zh-cn/docs/) 了解如何安装 Hexo

在 Hexo 中有两份主要的配置文件，其名称都是 \_config.yml。 其中，一份位于站点根目录下，主要包含 Hexo 本身的配置；另一份位于主题目录下，这份配置由主题作者提供，主要用于配置主题相关的选项。

为了描述方便，在以下说明中，将前者称为 站点配置文件， 后者称为 主题配置文件。

### 安装 NexT

Hexo 安装主题的方式非常简单，只需要将主题文件拷贝至站点目录的 themes 目录下， 然后修改下配置文件即可。具体到 NexT 来说，安装步骤如下。

#### 下载主题

如果你熟悉 [Git](http://git-scm.com/)， 建议你使用 克隆最新版本 的方式，之后的更新可以通过 git pull 来快速更新， 而不用再次下载压缩包替换。

* [克隆最新版本](http://theme-next.iissnan.com/getting-started.html#clone)
* [下载稳定版本](http://theme-next.iissnan.com/getting-started.html#stable)

在终端窗口下，定位到 Hexo 站点目录下。使用 Git checkout 代码：

$ cd your-hexo-site

$ git clone https://github.com/theme-next/hexo-theme-next themes/next

如果你对 Git 感兴趣，可以通过《Pro Git》这本书来学习。你可以访问我制作的一个 [在线版本（第一版）](http://iissnan.com/progit)。

#### 启用主题

与所有 Hexo 主题启用的模式一样。 当 克隆/下载 完成后，打开 站点配置文件， 找到 theme 字段，并将其值更改为 next。

启用 NexT 主题

theme: next

到此，NexT 主题安装完成。下一步我们将验证主题是否正确启用。在切换主题之后、验证之前， 我们最好使用 hexo clean 来清除 Hexo 的缓存。

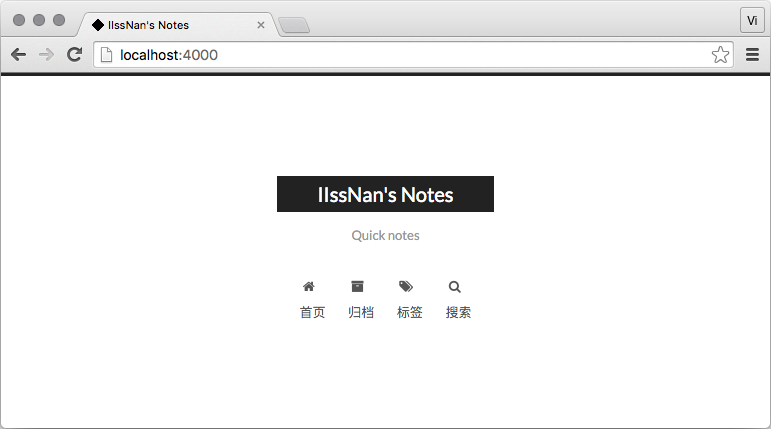
#### 验证主题

首先启动 Hexo 本地站点，并开启调试模式（即加上 --debug），整个命令是 hexo s --debug。 在服务启动的过程，注意观察命令行输出是否有任何异常信息，如果你碰到问题，这些信息将帮助他人更好的定位错误。 当命令行输出中提示出：

INFO Hexo is running at http://0.0.0.0:4000/. Press Ctrl+C to stop.

此时即可使用浏览器访问 http://localhost:4000，检查站点是否正确运行。

当你看到站点的外观与下图所示类似时即说明你已成功安装 NexT 主题。这是 NexT 默认的 Scheme —— Muse



现在，你已经成功安装并启用了 NexT 主题。下一步我们将要更改一些主题的设定，包括个性化以及集成第三方服务。

### 主题设定

#### 选择 Scheme

Scheme 是 NexT 提供的一种特性，借助于 Scheme，NexT 为你提供多种不同的外观。同时，几乎所有的配置都可以 在 Scheme 之间共用。目前 NexT 支持三种 Scheme，他们是：

* Muse - 默认 Scheme，这是 NexT 最初的版本，黑白主调，大量留白
* Mist - Muse 的紧凑版本，整洁有序的单栏外观
* Pisces - 双栏 Scheme，小家碧玉似的清新

Scheme 的切换通过更改 主题配置文件，搜索 scheme 关键字。 你会看到有三行 scheme 的配置，将你需用启用的 scheme 前面注释 # 去除即可。

选择 Pisces Scheme

#scheme: Muse

#scheme: Mist

scheme: Pisces

#### 设置 语言

编辑 站点配置文件， 将 language 设置成你所需要的语言。建议明确设置你所需要的语言，例如选用简体中文，配置如下：

language: zh-Hans

目前 NexT 支持的语言如以下表格所示：

| **语言** | **代码** | **设定示例** |
| --- | --- | --- |
| English | en | language: en |
| 简体中文 | zh-Hans | language: zh-Hans |
| Français | fr-FR | language: fr-FR |
| Português | pt | language: pt or language: pt-BR |
| 繁體中文 | zh-hk 或者 zh-tw | language: zh-hk |
| Русский язык | ru | language: ru |
| Deutsch | de | language: de |
| 日本語 | ja | language: ja |
| Indonesian | id | language: id |
| Korean | ko | language: ko |

#### 设置 菜单

菜单配置包括三个部分，第一是菜单项（名称和链接），第二是菜单项的显示文本，第三是菜单项对应的图标。 NexT 使用的是 [Font Awesome](http://fontawesome.io/) 提供的图标， Font Awesome 提供了 600+ 的图标，可以满足绝大的多数的场景，同时无须担心在 Retina 屏幕下 图标模糊的问题。

编辑 主题配置文件，修改以下内容：

1. 设定菜单内容，对应的字段是 menu。 菜单内容的设置格式是：item name: link。其中 item name 是一个名称，这个名称并不直接显示在页面上，她将用于匹配图标以及翻译。

菜单示例配置

menu:

home: /

archives: /archives

#about: /about

#categories: /categories

tags: /tags

#commonweal: /404.html

若你的站点运行在子目录中，请将链接前缀的 / 去掉

NexT 默认的菜单项有（标注 的项表示需要手动创建这个页面）：

| **键值** | **设定值** | **显示文本（简体中文）** |
| --- | --- | --- |
| home | home: / | 主页 |
| archives | archives: /archives | 归档页 |
| categories | categories: /categories | 分类页 |
| tags | tags: /tags | 标签页 |
| about | about: /about | 关于页面 |
| commonweal | commonweal: /404.html | 公益 404 |

1. 设置菜单项的显示文本。在第一步中设置的菜单的名称并不直接用于界面上的展示。Hexo 在生成的时候将使用 这个名称查找对应的语言翻译，并提取显示文本。这些翻译文本放置在 NexT 主题目录下的 languages/{language}.yml （{language} 为你所使用的语言）。

以简体中文为例，若你需要添加一个菜单项，比如 something。那么就需要修改简体中文对应的翻译文件 languages/zh-Hans.yml，在 menu 字段下添加一项：

menu:

home: 首页

archives: 归档

categories: 分类

tags: 标签

about: 关于

search: 搜索

commonweal: 公益404

something: 有料

1. 设定菜单项的图标，对应的字段是 menu\_icons。 此设定格式是 item name: icon name，其中 item name 与上一步所配置的菜单名字对应，icon name 是 Font Awesome 图标的 名字。而 enable 可用于控制是否显示图标，你可以设置成 false 来去掉图标。

菜单图标配置示例

menu\_icons:

enable: true

# Icon Mapping.

home: home

about: user

categories: th

tags: tags

archives: archive

commonweal: heartbeat

在菜单图标开启的情况下，如果菜单项与菜单未匹配（没有设置或者无效的 Font Awesome 图标名字） 的情况下，NexT 将会使用 作为图标。

请注意键值（如 home）的大小写要严格匹配

#### 设置 侧栏

默认情况下，侧栏仅在文章页面（拥有目录列表）时才显示，并放置于右侧位置。 可以通过修改 主题配置文件 中的 sidebar 字段来控制侧栏的行为。侧栏的设置包括两个部分，其一是侧栏的位置， 其二是侧栏显示的时机。

1. 设置侧栏的位置，修改 sidebar.position 的值，支持的选项有：
   * left - 靠左放置
   * right - 靠右放置

目前仅 Pisces Scheme 支持 position 配置。影响版本5.0.0及更低版本。

sidebar:

position: left

1. 设置侧栏显示的时机，修改 sidebar.display 的值，支持的选项有：
   * post - 默认行为，在文章页面（拥有目录列表）时显示
   * always - 在所有页面中都显示
   * hide - 在所有页面中都隐藏（可以手动展开）
   * remove - 完全移除
2. sidebar:
3. display: post

已知侧栏在 use motion: false 的情况下不会展示。 影响版本5.0.0及更低版本。

#### 设置 头像

编辑 主题配置文件， 修改字段 avatar， 值设置成头像的链接地址。其中，头像的链接地址可以是：

| **地址** | **值** |
| --- | --- |
| 完整的互联网 URI | http://example.com/avatar.png |
| 站点内的地址 | 将头像放置主题目录下的 source/uploads/ （新建 uploads 目录若不存在）  配置为：avatar: /uploads/avatar.png  或者 放置在 source/images/ 目录下  配置为：avatar: /images/avatar.png |

头像设置示例

avatar: http://example.com/avatar.png

#### 设置 作者昵称

编辑 站点配置文件， 设置 author 为你的昵称。

#### 站点描述

编辑 站点配置文件， 设置 description 字段为你的站点描述。站点描述可以是你喜欢的一句签名:)

### NexT标签

「标签」(Tag Plugin) 是 Hexo 提供的一种快速生成特定内容的方式。 例如，在标准 Markdown 语法中，我们无法指定图片的大小。这种情景，我们即可使用标签来解决。 Hexo 内置来许多标签来帮助写作者可以更快的书写， [完整的标签列表](https://hexo.io/docs/tag-plugins.html) 可以参考 Hexo 官网。 另外，Hexo 也开放来接口给主题，使主题有可能提供给写作者更简便的写作方法。 以下标签便是 NexT 主题当前提供的标签。

#### 文本居中的引用

此标签将生成一个带上下分割线的引用，同时引用内文本将自动居中。 文本居中时，多行文本若长度不等，视觉上会显得不对称，因此建议在引用单行文本的场景下使用。 例如作为文章开篇引用 或者 结束语之前的总结引用。

###### 使用方式

* HTML方式：使用这种方式时，给 img 添加属性 class="blockquote-center" 即可。
* 标签方式：使用 centerquote 或者 简写 cq。

此标签要求 NexT 的版本在 0.4.5 或以上。 若你正在使用的版本比较低，可以选择使用 HTML 方式。

标签调用方法

<!-- HTML方式: 直接在 Markdown 文件中编写 HTML 来调用 -->

<!-- 其中 class="blockquote-center" 是必须的 -->

<blockquote class="blockquote-center">blah blah blah</blockquote>

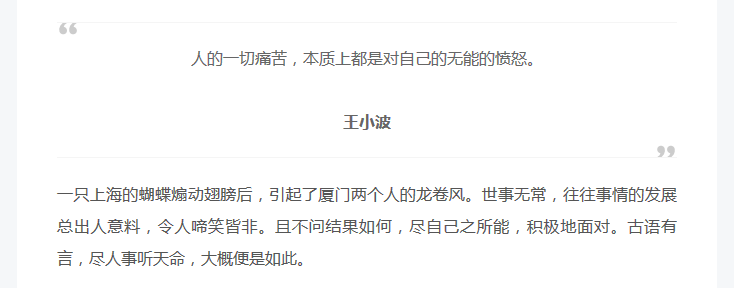
<!-- 标签 方式，要求版本在0.4.5或以上 -->

{% centerquote %}blah blah blah{% endcenterquote %}

<!-- 标签别名 -->

{% cq %} blah blah blah {% endcq %}

###### 效果示例:



#### 突破容器宽度限制的图片

当使用此标签引用图片时，图片将自动扩大 26%，并突破文章容器的宽度。 此标签使用于需要突出显示的图片, 图片的扩大与容器的偏差从视觉上提升图片的吸引力。 此标签有两种调用方式（详细参看底下示例）：

###### 使用方式

* HTML方式：使用这种方式时，为 img 添加属性 class="full-image"即可。
* 标签方式：使用 fullimage 或者 简写 fi， 并传递图片地址、 alt 和 title 属性即可。 属性之间以逗号分隔。

此标签要求 NexT 的版本在 0.4.5 或以上。 若你正在使用的版本比较低，可以选择使用 HTML 方式。  
如果要在图片下显示图片的标题，请使用 标签方式 并给定 title 属性。

标签调用方法

<!-- HTML方式: 直接在 Markdown 文件中编写 HTML 来调用 -->

<!-- 其中 class="full-image" 是必须的 -->

<img src="/image-url" class="full-image" />

<!-- 标签 方式，要求版本在0.4.5或以上 -->

{% fullimage /image-url, alt, title %}

<!-- 别名 -->

{% fi /image-url, alt, title %}

###### 效果示例:



#### Bootstrap Callout 由 [ivan-nginx](https://github.com/iissnan/hexo-theme-next/pull/1160) 贡献

这些样式出现在 [Bootstrap 的官方文档](http://getbootstrap.com/)中。

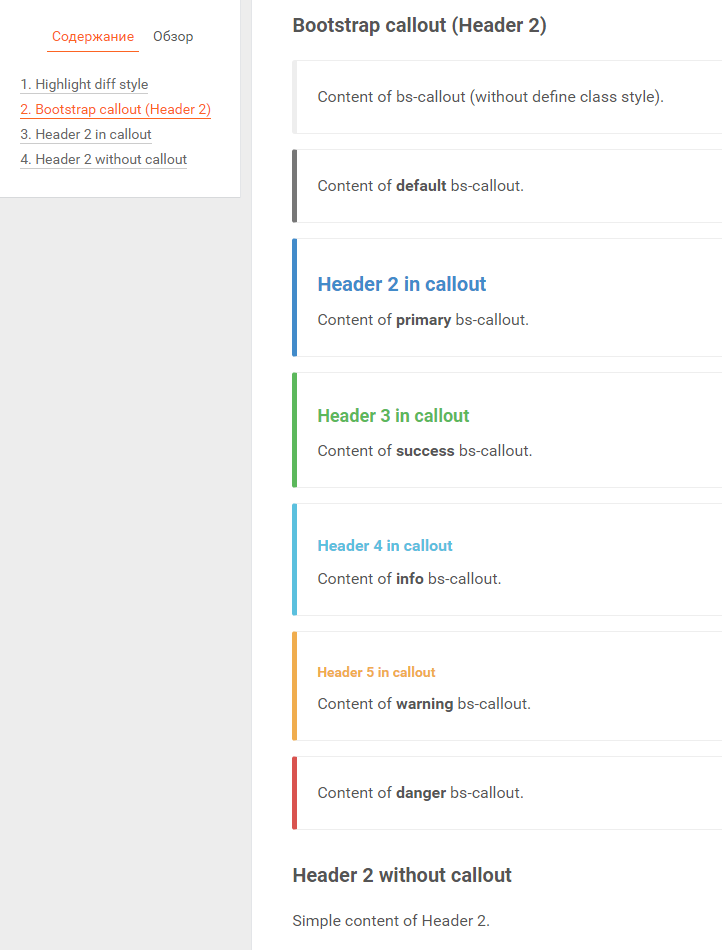
###### 使用方式

{% note class\_name %} Content (md partial supported) {% endnote %}

其中，class\_name 可以是以下列表中的一个值：

* default
* primary
* success
* info
* warning
* danger

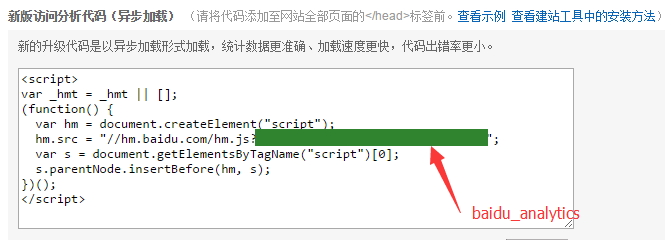
###### 效果示例:



##### 集成常用的第三方服务

###### 百度统计

注意： baidu\_analytics 不是你的百度 id 或者 百度统计 id

1. 登录 [百度统计](http://tongji.baidu.com/)， 定位到站点的代码获取页面
2. 复制 hm.js? 后面那串统计脚本 id，如： 
3. 编辑 主题配置文件， 修改字段 baidu\_analytics 字段，值设置成你的百度统计脚本 id。

###### 阅读次数统计（LeanCloud） 由 [Doublemine](https://github.com/iissnan/hexo-theme-next/pull/439) 贡献

请查看 [为NexT主题添加文章阅读量统计功能](https://notes.wanghao.work/2015-10-21-%E4%B8%BANexT%E4%B8%BB%E9%A2%98%E6%B7%BB%E5%8A%A0%E6%96%87%E7%AB%A0%E9%98%85%E8%AF%BB%E9%87%8F%E7%BB%9F%E8%AE%A1%E5%8A%9F%E8%83%BD.html#%E9%85%8D%E7%BD%AELeanCloud)

###### Algolia 搜索

详细的配置请参考： [第三方服务 - Algolia](http://theme-next.iissnan.com/third-party-services.html#algolia-search)

## 主题 Ayer

### 介绍

[Ayer](https://github.com/Shen-Yu/hexo-theme-ayer) 是一个干净且优雅的Hexo主题，自带响应式，加载速度很快，该有的功能都有，可配置项很多，非常适合作为你的博客主题，主题内还附送了6张精美的高清壁纸。欢迎使用和Star支持，如果你在使用过程中有任何疑问或者建议，欢迎联系我！如果你的博客采用了本主题，欢迎在下面评论区留下地址~

[Ayer](https://github.com/Shen-Yu/hexo-theme-ayer) 在马来语中是“水”的意思，在西班牙语中是“昨天”的意思。



[GitHub地址](https://github.com/Shen-Yu/hexo-theme-ayer)

[国内镜像](https://gitee.com/shen-yu/hexo-theme-ayer)

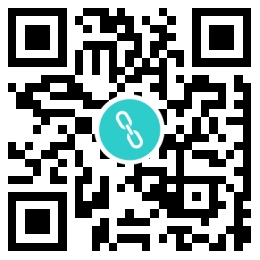
[效果预览](https://shen-yu.gitee.io/)

[中文说明](https://shen-yu.gitee.io/2019/ayer/)

### 特性

1. 干净且优雅，文章内容美观易读
2. 首页封面全屏平铺，扁平化设计，简洁又不失高大上
3. 响应式设计，博客在桌面端、平板、手机等设备上均能很好的展现
4. 首页封面和文案可以任意更换，主题内附送6张精美高清壁纸
5. 时间轴式的归档页
6. 支持 黑夜阅读 模式
7. 侧边栏可以点击显示或隐藏
8. 支持文章置顶和文章打赏
9. 支持文章字数和阅读时长统计
10. 支持网易云音乐播放
11. 支持 MathJax 和 Katex 数学公式
12. TOC 目录在文章页悬浮，跳转更方便
13. 可设置阅读文章时做密码验证[hexo-blog-encrypt](https://github.com/MikeCoder/hexo-blog-encrypt/blob/master/ReadMe.zh.md)
14. [Valine](https://valine.js.org/)和[Gitalk](https://gitalk.github.io/" \t "_blank)评论模块（推荐配合leancloud使用 Valine）
15. 集成了不蒜子、百度统计、Google Analytics、CNZZ等统计功能
16. 支持Github Ribbons

如果想体验手机端浏览效果，可以扫一下二维码：



### 安装

|  |  |
| --- | --- |
| 1 2 | COPY  # 国内用户如果速度较慢，可以把github地址替换为：https://gitee.com/mirrors/ayer.git git clone https://github.com/Shen-Yu/hexo-theme-ayer.git themes/ayer |

### 修改

将博客根目录下的 \_config.yml 里的 theme 值修改成 ayer

|  |  |
| --- | --- |
| 1 | COPY  theme: ayer |

### 更新

更新主题可能会覆盖你之前改动过的config文件，建议先把config.yml里的配置备份一下，然后删掉ayer目录后重新git clone下载主题，再把配置复制进去。

|  |  |
| --- | --- |
| 1 2 | COPY  cd themes/ayer git pull |

### 主题配置

以下是ayer主题目录下的 \_config.yml文件配置，请先确保你已经读过[Hexo文档](https://hexo.io/zh-cn/docs/" \t "_blank)。如果你在配置过程中有问题，请先擅用 [搜索引擎](https://cn.bing.com/)。如果你搜不到解决方法，你可以参考：[我的博客源码](https://gitee.com/shen-yu/shen-yu/tree/dev/)，按照我的一模一样配肯定是不会有问题的。如果对部分字体或颜色等有定制需求，请自行在css文件进行修改，主题本身已经很符合审美，自定义配置项也很多，不建议改得花里胡哨，过大的字体和库都会影响博客加载速度。如果还有问题或者建议，那么请在评论区给我留言~

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 | COPY  # 侧边栏菜单 menu:  主页: /  归档: /archives  分类: /categories  标签: /tags  摄影: http://shenyu-vip.lofter.com  旅行: /tags/旅行/  关于我: /2019/about  # 站点次标题和打字动效 # https://github.com/mattboldt/typed.js subtitle:  enable: true # 是否开启动效  text: 面朝大海，春暖花开 # 显示的文字  text2: 愿你一生努力，一生被爱 # 滚动播放，如果不需要可以留空  text3: 想要的都拥有，得不到的都释怀 # 最多支持三段文字  startDelay: 0 # 延迟时间  typeSpeed: 200 # 打字速度  loop: true # 是否循环  backSpeed: 100 # 回退速度  showCursor: true # 是否显示光标  # 网站图标和侧边栏logo favicon: /favicon.ico logo: /images/ayer-side.svg  # 封面配置 # enable-是否启用封面；path-封面背景图；logo-封面logo cover:  enable: true  path: /images/cover1.jpg # /source/images目录下附送多张精美壁纸，可任意更换  logo: /images/ayer.svg # 如果不要直接设置成false  # 页面顶部进度条  progressBar: true  # 网易云音乐插件 music:  enable: false  # 播放器尺寸类型(1：小尺寸、2：大尺寸)  type: 1  id: 518895142 # 网易云分享的音乐ID(更换音乐请更改此配置项)  autoPlay: true # 是否开启自动播放  # 文章配置 # 文章太长，截断按钮文字(在需要截断的行增加此标记：<!--more-->) excerpt\_link: 阅读更多... # 如果你嫌每篇文章手动加more标记比较麻烦，又不想在首页全文显示，可以把excerpt\_all设置成true，这样首页只会显示文章归档 excerpt\_all: false  # 是否开启代码复制按钮 copy\_btn: true # 是否开启文章分享按钮 share\_enable: true # 国内的社交平台(If you are not in China, maybe you prefer to set:false) share\_china: true # 文章分享文字 share\_text: 分享  # 分页文字 nav\_text:  page\_prev: 上一页  page\_next: 下一页  post\_prev: 上一篇  post\_next: 下一篇  # 文章页是否显示目录 toc: true  # 文章中的图片是否支持点击放大 image\_viewer: true  # https://github.com/willin/hexo-wordcount # 是否开启字数统计(关闭请设置enable为false) # 也可以单独在md文件里Front-matter设置`no\_word\_count: true`属性，来自定义关闭字数统计 word\_count:  enable: true  # 只在文章详情显示(不在首页显示)  only\_article\_visit: true  # 打赏 # 打赏type设定：0-关闭打赏； 1-文章对应的md文件里有reward:true属性，才有打赏； 2-默认开启所有文章均有打赏，如果有些文章你不需要，请在文章对应的md文件里设置no\_reward:true reward\_type: 1 # 打赏wording reward\_wording: '请我喝杯咖啡吧~' # 支付宝二维码图片地址，跟你设置logo的方式一样。比如：/images/alipay.jpg alipay: /images/alipay.jpg # 微信二维码图片地址 weixin: /images/wechat.jpg  # 版权声明 # 版权声明type设定：0-关闭版权声明； 1-文章对应的md文件里有copyright: true属性，才有版权声明； 2-所有文章均有版权声明 copyright\_type: 2  # 是否启用搜索 search: true  # RSS订阅(先安装hexo-generator-feed插件，再去博客根目录config进行配置) # 不想显示可以直接留空 rss: /atom.xml  # 是否启用黑夜模式开关 darkmode: true  # 动态背景效果: 0-关闭，1-动态线条(跟随鼠标) canvas\_bg: 0  # 自定义鼠标样式，直接替换/images/mouse.cur文件 mouse:  enable: false  path: /images/mouse.cur  # 鼠标点击效果：0-关闭，1-爱心，2-爆炸烟花，3-粒子烟花 click\_effect: 0  # 页面宽度自定义（不建议修改，可能造成布局混乱），article\_width文章宽度，sidebar\_width侧边栏宽度 layout:  article\_width: 80rem  sidebar\_width: 8rem   # 评论：1、Valine(推荐)；2、Gitalk  # 1、Valine[一款快速、简洁且高效的无后端评论系统](https://github.com/xCss/Valine) # 启用Valine必须先创建leancloud应用， 注册账号后获取 id|key 填入即可 leancloud:   enable: true  app\_id: #  app\_key: # # Valine配置(如果有些文章你想关闭评论，请在文章对应的md文件里设置no\_valine:true) valine:  enable: true # 是否启用  verify: false # 是否开启防垃圾评论验证  notify: false # 是否开启邮件提醒(https://valine.js.org/notify.html)  avatar: mp # 头像样式(https://valine.js.org/avatar.html)  placeholder: 给我的文章加点评论吧~ # placeholder  # 2、Gitalk(https://github.com/gitalk/gitalk) gitalk:  enable: false # true  clientID: # GitHub Application Client ID  clientSecret: # Client Secret  repo: # Repository name  owner: # GitHub ID  admin: # GitHub ID   # GitHub Ribbons-封面右上角的forkme，换样式直接在source/images目录下替换forkme图片 github:   # (关闭请设置为false)  url: https://github.com/Shen-Yu/hexo-theme-ayer  # fancybox(仅用于相册展示，若需要可配置albums) fancybox: true  # 访问量统计(不蒜子) busuanzi:  enable: true  # 友盟cnzz统计(url填js代码src链接) cnzz:  enable: true  url: #  # Google Analytics google\_analytics: '' # 百度统计 baidu\_analytics: ''  # Mathjax数学公式 mathjax: true  # Katex数学公式(allpost设置为false时只有头部设置math:true的文章才开启) # 需要更换hexo渲染器，npm un hexo-renderer-marked -S && npm i hexo-renderer-markdown-it-katex -S katex:  enable: false # true  allpost: true  copy\_tex: false  # 网站成立年份(默认为 2019，若填入年份小于当前年份，则显示为 2018-2019 类似的格式) since: 2019  #是否显示页脚信息(建议保留，有助于本主题的推广) pageFooter: true  #ICP备案信息尾部显示 icp:  enable: false  url: 'http://www.beian.miit.gov.cn/' # 备案链接  text: '浙ICP备88888888' # 备案信息 |

#### 插件(必需)

* [hexo-generator-searchdb](https://github.com/theme-next/hexo-generator-searchdb) 用于搜索

|  |  |
| --- | --- |
| 1 | COPY  npm install hexo-generator-searchdb --save |

* 然后将以下配置复制到你博客根目录下的 \_config.yml 里（注意不是ayer目录下的）:

|  |  |
| --- | --- |
| 1 2 3 4 5 | COPY  # hexo-generator-searchdb search:  path: search.xml  field: post  format: html |

* [hexo-generate-feed](https://github.com/hexojs/hexo-generator-feed) 用于生成RSS订阅

|  |  |
| --- | --- |
| 1 | COPY  npm install hexo-generator-feed --save |

* 然后将以下配置复制到你博客根目录下的 \_config.yml 里（注意不是ayer目录下的）:

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | COPY  feed:  type: atom  path: atom.xml  limit: 20  hub:  content:  content\_limit: 140  content\_limit\_delim: ' '  order\_by: -date |

#### 插件(可选)

* [hexo-generator-index-pin-top](https://github.com/netcan/hexo-generator-index-pin-top) 用于文章置顶
* [hexo-blog-encrypt](https://github.com/MikeCoder/hexo-blog-encrypt/blob/master/ReadMe.zh.md) 用于文章加密
* [hexo-tag-aplayer](https://github.com/MoePlayer/hexo-tag-aplayer/blob/master/docs/README-zh_cn.md) 用于播放音乐
* [hexo-tag-dplayer](https://github.com/MoePlayer/hexo-tag-dplayer) 用于播放视频
* [hexo-helper-live2d](https://github.com/EYHN/hexo-helper-live2d/blob/master/README.zh-CN.md) 二次元看板娘

更多插件请见 [hexo插件市场](https://hexo.io/plugins/" \t "_blank)

#### 分类

|  |  |
| --- | --- |
| 1 | COPY  hexo new page categories |

然后将以下复制到/source/categories/index.md文件

|  |  |
| --- | --- |
| 1 2 3 4 5 | COPY  --- title: categories type: "categories" layout: "categories" --- |

#### 标签

|  |  |
| --- | --- |
| 1 | COPY  hexo new page tags |

配置同分类一样

#### 相册

|  |  |
| --- | --- |
| 1 | COPY  hexo new page photos |

然后将以下复制到/source/photos/index.md文件，img\_url替换成图片路径，caption替换成图片名称

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 | COPY  --- title: Gallery  albums: [  ["img\_url","img\_caption"],  ["img\_url","img\_caption"]  ] --- |

#### 文章目录

用Tocbot解析文章标题并生成目录

* 将以下配置复制到你ayer目录下的 \_config.yml 里：

|  |  |
| --- | --- |
| 1 2 | COPY  # Toc  toc: true |

* 当然你可能并不想所有文章都生成悬浮目录，你可以在文章顶部的配置中加一行来进行关闭：

|  |  |
| --- | --- |
| 1 2 3 | COPY  --- no\_toc: true --- |

### 常见问题

#### 一.本地图片引用了却无法显示

插入图片的两种方法：1.引用图床，2.引用本地图片。为了防止路径出错，建议使用图床。

##### 1.图床

推荐使用：[SM.MS](https://sm.ms/" \t "_blank)，[聚合图床](https://www.superbed.cn/" \t "_blank)

##### 2.本地图片

建议在hexo创建的博客目录下的source中建立一个images文件夹专门放图片，路径：你的博客目录/source/images，如果还是无法显示，请尝试在文章里用html的img标签来引用本地图片。

#### 二.可以启动但是样式变形了

这很可能是因为你把博客部署到了服务器子目录，导致静态资源加载的路径不对，在此强烈建议你部署到根目录，github和码云都支持根目录部署，只要你把仓库名搞成和用户名一样就行了。

#### 三.点击侧边栏页面无法显示

原因：ayer主题目录下的 config.yml 里menu属性的路径不对，这需要你根据页面路径手动配置。

以我的博客为例：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 | COPY  menu:  主页: /  归档: /archives  分类: /categories  标签: /tags  旅行: /tags/旅行/  摄影: http://shenyu-vip.lofter.com  关于我: /2019/about |

上面的归档/archives是固定的，你不需要修改，标签和分类都需要手动创建目录，旅行实际上就是一个标签页，当你在文章头部设置tags属性后就会自动生成，摄影是一个站外的绝对路径，关于我实际上就是一个正常的文章页。好了，现在一个侧边栏导航就已经配置好了。

#### 四.为什么修改了配置却没有效果

建议每次生成站点或部署之前都用 hexo clean 命令清理一下缓存，另外你可能还需要刷新一下。

#### 五.怎么修改大标题和站点信息

站点配置文件\_config.yml是hexo站点根目录下的主配置文件(还不知道是哪里的，自己google)，注意：别和ayer主题目录下的\_config.yml搞混了。

修改网站各种资料，例如标题、副标题和邮箱等个人资料，请修改站点根目录下的\_config.yml

以我的博客为例：

#### 六.怎么添加百度或谷歌统计

##### 百度统计

在ayer的config配置里的baidu\_analytics填上百度统计代码中的hm.js问好后面那一串东西

##### 谷歌统计

在ayer的config配置里的google\_analytics填上谷歌统计代码的跟踪ID，即UA值(包含UA)

#### 七.怎么查看文章的访问量

在ayer的config配置里有busuanzi选项，默认设置为enable:true，就开启了站点访问量的统计，站点尾部左下角的PV代表了当前页的访问次数，UV代表了当前站点的访问人数。注意：不蒜子统计是根据当前域名来统计的，你在本地启动的时候由于是localhost，所以数字会很大，部署到线上之后就会根据域名重新统计。

#### 八.网易云音乐无法播放

这是由于网易云音乐对部分歌曲做了版权限制，你可以多换几首试试，配置文件里的默认歌曲 潮鳴り 是可以播放的。如果你动手能力强，也可以试试音乐播放插件 [hexo-tag-aplayer](https://github.com/MoePlayer/hexo-tag-aplayer/blob/master/docs/README-zh_cn.md" \t "_blank)，配合MetingJS可以支持对于 QQ音乐、网易云、虾米、酷狗、百度等平台的音乐播放。

#### 九.分享链接无法打开

你需要在根目录下的\_config.yml文件中把url配置成部署后的域名。

#### 十.怎么让搜索引擎收录站点

参考：[Hexo站点SEO优化技巧](https://shen-yu.gitee.io/2020/hexo-seo)

## 主题hexo icarus

### 安装icarus

* 直接下载主题模块放到blog项目 ,blog项目根目录执行

git clone https://github.com/ppoffice/hexo-theme-icarus.git themes/icarus

此时已经下载到项目中。

* 顶级\_config.yml中选择icarus主题
* # Extensions
* ## Plugins: https://hexo.io/plugins/
* ## Themes: https://hexo.io/themes/
* theme: icarus
* 此时主题已经安装好，清除、编译、部署可以看到效果了

在博客目录中输入

git clone https://github.com/ppoffice/hexo-theme-icarus.git themes/icarus

修改 \_config.yml 内的 theme:icarus 设定，即可切换主题

### 配置icarus

* 完全参照[官网](https://blog.zhangruipeng.me/hexo-theme-icarus/categories/Configuration/Posts/)配置，进行翻译解说

#### 配置 icarus 主题

下载 [icarus](https://github.com/ppoffice/hexo-theme-icarus/releases" \t "_blank) ，在 hexo 的 themes 文件夹，新加入 [icarus](https://github.com/ppoffice/hexo-theme-icarus" \t "_blank) 目录，配置 hexo 的根目录的 \_config.yml

\_config.yml

|  |  |
| --- | --- |
| 1 2 | -theme: yilia +theme: icarus |

然后更新部署就好了

|  |  |
| --- | --- |
| 1 | hexo d -g |

#### 配置文章部分

#### 顶部图片添加

icarus 主题中的配置\_config.yml中开启图片开关

article:

thumbnail: true

文章.md文件头中添加图片绝对/相对地址

title: Getting Started with Icarus

thumbnail: /gallery/thumbnails/desert.jpg

// thumbnail:https://raw.githubusercontent.com/removeif/blog\_image/master/20190620152744.png

---

Post content...

配置完成后部署显示效果 (最新文章列表显示缩略图、文章开头显示一张设置图片)

#### 左边文章导航栏开启

icarus 主题中的配置\_config.yml中开关

widgets:

-

type: toc

position: left

#### 提取文章目录

title: Table of Contents Example

toc: true

---

Post content...

#### 评论系统开启

icarus 主题中的配置\_config.yml中开启（部分评论系统需要翻墙才能使用，valine不用翻墙个人推荐，[valine安装参考](https://valine.js.org/quickstart.html)）

comment:

type: valine

app\_id: xxxxxxxx # (required) LeanCloud application id

app\_key: xxxxxxxx # (required) LeanCloud application key

notify: false # (optional) receive email notification

verify: false # (optional) show verification code

placeholder: xxxxxxxx # (optional) comment box placeholder text

#### 捐赠收款开启

icarus 主题中的配置\_config.yml中开启 注意如果默认不配置，编译时有报错，# 把它注释掉后会报错，不启用功能方法是注释掉代码

donate:

-

# Donation entry name

type: alipay

# Qrcode image URL

qrcode: 'https://wx2.sinaimg.cn/large/b5d1b710gy1g0lvxdcwm0j20p011i4bg.jpg'

-

# Donation entry name

type: wechat

# Qrcode image URL

qrcode: 'https://wx2.sinaimg.cn/large/b5d1b710gy1g0lvwdcpb5j20u014qgy2.jpg'

#### 全局搜索开启

icarus 主题中的配置\_config.yml中开启,不同的搜索类型需要[安装插件参考官网](https://blog.zhangruipeng.me/hexo-theme-icarus/categories/Plugins/Search/),type: insight此类型不需要安装，已经内置

search:

type: insight

#### 更多配置请参考官网配置

目前配置基本已经够使用，还需要更多配置请参考[连接](https://blog.zhangruipeng.me/hexo-theme-icarus/categories/Configuration/Posts/)

https://blog.zhangruipeng.me/hexo-theme-icarus/categories/Configuration/Posts/

### 自定义 icarus 主题

### 代码高亮

[代码高亮款式](https://github.com/highlightjs/highlight.js/tree/master/src/styles) , [预览参见这里](https://highlightjs.org/)，Android 程序员选择 androidstudio 效果不错

themes/icarus/\_config.yml

|  |  |
| --- | --- |
| 1 | highlight: androidstudio |

### 更新logo

icarus 有默认的 logo，这里想改变一下，logo 是在 themes\icarus\source\images 里面logo.svg 文件，我们只需要替换成自己的 logo 文件即可，可以从[这里](https://icomoon.io/app/" \l "/select" \t "_blank) 去获取。

### 页面类型判定

注: 可以通过变量 page.layout 来判断当前页面类型，'post' 表示当前是文章页面，具体可参考 [hexo](https://hexo.io/zh-cn/docs/variables" \t "_blank) 文档。

### 布局

#### 文章页面两栏布局

主题默认是三栏布局，并且显示了很多的 widget ，但在阅读文章时显得有些拥挤。因此在文章页面，修改为两栏布局，并显示特定的 widget。

diff:includes/helpers/layout.js

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | const widgets = hexo.extend.helper.get('get\_config').bind(this)('widgets'); - return widgets.filter(widget => widget.hasOwnProperty('position') && widget.position === position); + if (this.page.layout !== 'post') { + return widgets.filter(widget => widget.hasOwnProperty('position') && widget.position === position); + } + if (position === 'left') { + return widgets.filter(widget => widget.hasOwnProperty('position') && (widget.type === 'toc' || widget.type === 'profile')); + } else { + return [] + } |

可以参考上述代码，即可实现不同页面不同 widget。

但两栏整体宽度跟三栏不同。因此强制指定为三栏布局，并且修改相应的宽度，这样所有的页面侧边栏宽度保持一致。

diff:layout/common/widget.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | <% function side\_column\_class() {  switch (column\_count()) {  case 2: - return 'is-4-tablet is-4-desktop is-4-widescreen'; + return 'is-4-tablet is-4-desktop is-3-widescreen';  case 3: |

diff:layout/layout.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | -<body class="is-<%= column\_count() %>-column"> +<body class="is-3-column">  <%- partial('common/navbar', { page }) %>  <% function main\_column\_class() {  switch (column\_count()) {  case 1:  return 'is-12';  case 2: - return 'is-8-tablet is-8-desktop is-8-widescreen'; + return 'is-8-tablet is-8-desktop is-9-widescreen'; |

并修改在不同屏幕小大下的宽度

diff:source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | .is-2-column .container  max-width: screen-desktop - 2 \* gap  width: screen-desktop - 2 \* gap + .is-3-column .container + max-width: screen-widescreen - gap + width: screen-widescreen - gap  @media screen and (min-width: screen-fullhd) + .is-3-column .container + max-width: screen-fullhd - 2 \* gap + width: screen-fullhd - 2 \* gap  .is-2-column .container  max-width: screen-widescreen - 2 \* gap  width: screen-widescreen - 2 \* gap |

#### 调整阅读模式双栏

判断 post 页面，显示目录 toc ，修改宽度

themes/icarus/includes/helpers/layout.js

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | - return widgets.filter(widget => widget.hasOwnProperty('position') && widget.position === position); + if (this.page.layout !== 'post') { + return widgets.filter(widget => widget.hasOwnProperty('position') && widget.position === position); + } + if (position === 'right') { + return widgets.filter(widget => widget.hasOwnProperty('position') && (widget.type === 'toc')); + } else { + return [] + } |

themes/icarus/layout/common/widget.ejs

|  |  |
| --- | --- |
| 1 2 3 | case 2: - return 'is-4-tablet is-4-desktop is-4-widescreen'; + return 'is-4-tablet is-4-desktop is-3-widescreen'; |

themes/icarus/layout/layout.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | <head>  <%- partial('common/head') %>  </head> -<body class="is-<%= column\_count() %>-column"> +<body class="is-3-column">  <%- partial('common/navbar', { page }) %>  case 2: - return 'is-8-tablet is-8-desktop is-8-widescreen'; + return 'is-8-tablet is-8-desktop is-9-widescreen';  case 3:  return 'is-8-tablet is-8-desktop is-6-widescreen'  } |

themes/icarus/source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | .is-2-column .container  max-width: screen-desktop - 2 \* gap  width: screen-desktop - 2 \* gap + .is-3-column .container + max-width: screen-widescreen - gap + width: screen-widescreen - gap  @media screen and (min-width: screen-fullhd) + .is-3-column .container + max-width: screen-fullhd - 2 \* gap + width: screen-fullhd - 2 \* gap |

#### 优化文章标题布局

标题移动到文章信息上方，增加更新时间，并增加了icon。

diff:layout/common/article.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 | <div class="card-content article <%= post.hasOwnProperty('direction') ? post.direction : '' %>"> + <h1 class="title is-size-3 is-size-4-mobile has-text-weight-normal"> + <% if (index) { %> + <a class="has-link-black-ter" href="<%- url\_for(post.link ? post.link : post.path) %>"><i class="fas fa-angle-double-right"></i><%= post.title %></a> + <% } else { %> + <i class="fas fa-angle-double-right"></i><%= post.title %> + <% } %> + </h1>  <% if (post.layout != 'page') { %>  <div class="level article-meta is-size-7 is-uppercase is-mobile is-overflow-x-auto">  <div class="level-left"> - <time class="level-item has-text-grey" datetime="<%= date\_xml(post.date) %>"><%= date(post.date) %></time> + <time class="level-item has-text-grey" datetime="<%= date\_xml(post.date) %>"><i class="far fa-calendar-alt">&nbsp;</i><%= date(post.date) %></time> + <% if (post.updated && post.updated > post.date) { %> + <time class="level-item has-text-grey is-hidden-mobile" datetime="<%= date\_xml(post.updated) %>"><i class="far fa-calendar-check">&nbsp;</i><%= date(post.updated) %></time> + <% } %>  <% if (post.categories && post.categories.length) { %> |

其中创建时间使用日期。

diff:source/js/main.js

|  |  |
| --- | --- |
| 1 2 3 4 5 | - if (typeof(moment) === 'function') { - $('.article-meta time').each(function () { - $(this).text(moment($(this).attr('datetime')).fromNow()); - }); - } |

#### 优化文章结尾布局

在文章结尾增加一个 hr，并修改 tags 展示。在预览时也显示 tags，并且将 Read More 按钮放置在右边。

diff:layout/common/article.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 | <% if (!index && post.tags && post.tags.length) { %> + <hr style="height:1px;margin:1rem 0"/>  <div class="level is-size-7 is-uppercase">  <div class="level-start">  <div class="level-item"> - <span class="is-size-6 has-text-grey has-mr-7">#</span> + <i class="fas fa-tags has-text-grey"></i>&nbsp;  <%- list\_tags(post.tags, {  class: 'has-link-grey ',  show\_count: false, - style: 'link' + style: 'link', + separator: ',&nbsp;'  }) %>  </div>  </div>  </div>  <% } %>  <% if (index && post.excerpt) { %> - <div class="level is-mobile"> + <hr style="height:1px;margin:1rem 0"/> + <div class="level is-mobile is-flex"> + <div class="level-start"> + <% if (post.tags && post.tags.length) { %> + <div class="level-item is-size-7 is-uppercase"> + <i class="fas fa-tags has-text-grey"></i>&nbsp; + <%- list\_tags(post.tags, { + class: 'has-link-grey ', + show\_count: false, + style: 'link', + separator: ',&nbsp;' + }) %> + </div> + <% } %> + </div>  <div class="level-start"> |

#### 优化个人信息布局

减少头像大小，头像下方计数的地方增加链接，follow前增加icon。

diff:layout/widget/profile.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | - <nav class="level is-mobile"> + <nav class="level menu-list is-mobile" style="margin-bottom:1rem">  <div class="level-item has-text-centered is-marginless"> - <div> + <a href="<%- url\_for('/archives/') %>">  <p class="heading"> ...... |

#### 优化移动端显示

在移动端，隐藏 archive 和 tagcloud。

diff:layout/widget/archive.ejs

|  |  |
| --- | --- |
| 1 2 | -<div class="card widget"> +<div class="card widget is-hidden-mobile"> |

diff:layout/widget/tagcloud.ejs

|  |  |
| --- | --- |
| 1 2 | -<div class="card widget"> +<div class="card widget is-hidden-mobile"> |

#### 目录粘性布局

增加 column-left is-sticky 类。

diff:layout/widget/toc.ejs

|  |  |
| --- | --- |
| 1 2 | -<div class="card widget" id="toc"> +<div class="card widget column-left is-sticky" id="toc"> |

### 功能

#### 增加版权说明

diff:layout/common/article.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | <div class="content">  <%- index && post.excerpt ? post.excerpt : post.content %>  </div> + <% if (!index && post.layout === 'post' && post.copyright !== false) { %> + <ul class="post-copyright"> + <li><strong>本文标题：</strong><a href="<%= post.permalink %>"><%= page.title %></a></li> + <li><strong>本文作者：</strong><a href="<%= theme.url %>"><%= theme.author %></a></li> + <li><strong>本文链接：</strong><a href="<%= post.permalink %>"><%= post.permalink %></a></li> + <li><strong>发布时间：</strong><%= post.date.format("YYYY-MM-DD") %></li> + <li><strong>版权声明：</strong>本博客所有文章除特别声明外，均采用 <a href="https://creativecommons.org/licenses/by-nc-sa/4.0/deed.zh" rel="external nofollow" target="\_blank">CC BY-NC-SA 4.0</a> 许可协议。转载请注明出处！ + </li> + </ul> + <% } %>  <% if (!index && post.tags && post.tags.length) { %> |

并增加样式

diff:source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 | +.post-copyright + font-size: 1rem + letter-spacing: 0.02rem + word-break: break-all + margin: 2.5rem 0 0 + padding: 1rem 1rem + border-left: 3px solid #FF1700 + background-color: #F9F9F9 |

#### 增加文章版权

themes/icarus/layout/common/article.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | <div class="content">  <%- index && post.excerpt ? post.excerpt : post.content %>  </div> + <% if (!index && post.layout === 'post' && post.copyright !== false) { %> + <ul class="post-copyright"> + <li><strong>本文标题：</strong><a href="<%= post.permalink %>"><%= page.title %></a></li> + <li><strong>本文作者：</strong><a href="<%= theme.url %>"><%= theme.author %></a></li> + <li><strong>本文链接：</strong><a href="<%= post.permalink %>"><%= post.permalink %></a></li> + <li><strong>发布时间：</strong><%= post.date.format("YYYY-MM-DD") %></li> + <li><strong>版权声明：</strong>本博客所有文章除特别声明外，均采用 <a href="https://creativecommons.org/licenses/by-nc-sa/4.0/deed.zh" rel="external nofollow" target="\_blank">CC BY-NC-SA 4.0</a> 许可协议。转载请注明出处！ + </li> + </ul> + <% } %>  <% if (!index && post.tags && post.tags.length) { %> |

下面文件加入格式

themes/icarus/source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | border: none  .file  all: initial + +.post-copyright + font-size: 1rem + letter-spacing: 0.02rem + word-break: break-all + margin: 2.5rem 0 0 + padding: 1rem 1rem + border-left: 3px solid #FF1700 + background-color: #F9F9F9 |

这里大部分是参照 [自定义icarus](https://www.alphalxy.com/2019/03/customize-icarus/) 做的修改(点赞)

#### 配置 gitalk 评论

icarus 默认是支持 gitalk 的，所以只需要在 themes/icarus/\_config.yml 中设置一下即可

themes/icarus/\_config.yml

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 | comment:  # Name of the comment plugin  type: gitalk  owner: cloudy-liu # (required) GitHub user name  repo: cloudy-liu.github.io # (required) GitHub repository name  client\_id: xxxx # (required) OAuth application client id  client\_secret: xxx # (required) OAuth application client secret  admin: cloudy-liu |

还没配置的可以[参考这篇](https://www.liuyun.fun/2018/07/14/Hexo%E6%9B%B4%E6%8D%A2%E4%B8%BAGitTalk%E8%AF%84%E8%AE%BA%E7%B3%BB%E7%BB%9F/)做配置

#### 增加标题自动计数

diff:source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | +.article {counter-reset:section} +.article h2{counter-reset:sub-section} +.article h3{counter-reset:composite} +.article h4{counter-reset:detail} +.article h2:before{content:counter(section) " ";counter-increment:section} +.article h3:before{content:counter(section) "." counter(sub-section) " ";counter-increment:sub-section} +.article h4:before{content:counter(section) "." counter(sub-section) "." counter(composite) " ";counter-increment:composite} |

#### 默认显示目录

正常开启目录需要在 meta 信息中加入 toc: true 开启，但绝大部分文章都是有目录的，因此修改为默认开启。因为有多个地方引用了 get\_config('toc') 并且需要判断当前页面，改起来稍微麻烦一些，因此直接修改了 get\_config 方法。

diff:includes/helpers/config.js

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 | return defaultValue;  } else {  const property = readProperty(specs, configName); - return property === null ? null : property[descriptors.defaultValue]; + const result = property === null ? null : property[descriptors.defaultValue]; + return (configName === 'toc' && this.page.layout === 'post' && result === null) ? true : result;  } |

#### 页面footer显示一组icon

默认情况下一个icon对应一个链接，但例如 CC BY-NC-SA 4.0 需要四个图标一组。因此修改代码，使得配置 link.icon可以是一个数组，效果可以参考页面底部。

diff:layout/common/footer.ejs

|  |  |
| --- | --- |
| 1 2 3 4 | <% } else { %> - <i class="<%= link.icon %>"></i> + <% for (let icon of (Array.isArray(link.icon) ? link.icon : [link.icon])) { %><i class="<%= icon %>"></i>&nbsp;<% } %>  <% } %> |

diff:includes/specs/icon\_link.spec.js

|  |  |
| --- | --- |
| 1 2 3 4 5 | icon: {  [required]: true, - [type]: 'string', + [type]: ['string', 'array'],  [doc]: 'Link icon class names' |

\_config.yml 中配置如下

\_config.yml

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | footer:  links:  CC BY-NC-SA 4.0:  icon:  - fab fa-creative-commons  - fab fa-creative-commons-by  - fab fa-creative-commons-nc  - fab fa-creative-commons-sa  url: 'https://creativecommons.org/licenses/by-nc-sa/4.0/' |

#### 可以配置文章开头是否显示图片

Icarus 支持文章设置一个图片，在文章开头、最近的文章、时间线等地方显示。但有些图放大之后会显得很不协调，因此修改以支持自定义是否显示。修改 has\_thumbnail 方法，增加参数 isArticle 参数，默认 false，并在文章页面修改调用参数。

diff:includes/helpers/page.js

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 | - hexo.extend.helper.register('has\_thumbnail', function (post) { + hexo.extend.helper.register('has\_thumbnail', function (post, isArticle = false) {  const getConfig = hexo.extend.helper.get('get\_config').bind(this);  const allowThumbnail = getConfig('article.thumbnail', true);  if (!allowThumbnail) {  return false;  } + if (isArticle && post['article-thumbnail'] === false){ + return false; + }  return post.hasOwnProperty('thumbnail') && post.thumbnail; |

diff:layout/common/article.ejs

|  |  |
| --- | --- |
| 1 2 3 4 | <div class="card"> - <% if (has\_thumbnail(post)) { %> + <% if (has\_thumbnail(post, true)) { %>  <div class="card-image"> |

这样修改之后，如果文章 meta 信息中包含 article-thumbnail: false，就可以取消图片的显示。

#### ShareThis

**安装指南** [在线预览](https://blog.zhangruipeng.me/hexo-theme-icarus/uncategorized/sharethis-share-buttons/)

1. 访问[ShareThis](https://sharethis.com/)并点击页面上的“从分享按钮开始”(Start with Share Buttons)按钮。
2. 在“选择分享按钮类型”(Choose type of sharing button)页面选择你需要的按钮类型。 不要启用”GDPR规范工具“，否则可能会导致一些问题。 你也可以点击“自定义你的分享按钮”(Customize your share buttons)链接来进行按钮的进一步配置。 完成后点击“下一步”(Next)。

[选择按钮类型 - ShareThis](https://blog.zhangruipeng.me/hexo-theme-icarus/gallery/screenshots/sharethis-choose-button-type.png)

1. 在”选择你的CMS平台“(Choose your CMS platform)页面选择”HTML“并点击”下一步“(Next)。

[选择平台 - ShareThis](https://blog.zhangruipeng.me/hexo-theme-icarus/gallery/screenshots/sharethis-choose-platform.png)

1. 在“注册并获取代码”(Register and get the code!)页面输入邮箱和密码完成ShareThis的注册。
2. 最后，从HTML代码段中复制src中的URL地址到分享按钮配置中。

[获取代码 - ShareThis](https://blog.zhangruipeng.me/hexo-theme-icarus/gallery/screenshots/sharethis-get-code.png)

例如，下面的ShareThis代码：

AddThis代码

|  |  |
| --- | --- |
| 1 | <script type="text/javascript" src="https://platform-api.sharethis.com/js/sharethis.js#property=xxxxxxxxxxxxx&product=inline-share-buttons" async="async"></script> |

对应下面的主题配置：

themes/icarus/\_config.yml

|  |  |
| --- | --- |
| 1 2 3 | share:  type: sharethis  install\_url: https://platform-api.sharethis.com/js/sharethis.js#property=xxxxxxxxxxxxx&product=inline-share-buttons |

文章内容有误？请点击[此处](https://github.com/ppoffice/hexo-theme-icarus/edit/site/source/_posts/zh-CN/Share-Buttons.md" \t "_blank)提交修改。

#### 总访问量统计

icarus 是用卜算子统计，默认只统计了访问人数，并没有访问量统计，需要在页脚修改下

themes/icarus/layout/common/footer.ejs

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 | @@ -15,9 +15,8 @@  href="https://github.com/ppoffice/hexo-theme-icarus" target="\_blank">Icarus</a>  <% if (has\_config('plugins.busuanzi') ? get\_config('plugins.busuanzi') : false) { %>  <br> - <span id="busuanzi\_container\_site\_uv"> - <%- \_p('plugin.visitor', '<span id="busuanzi\_value\_site\_uv">0</span>') %> - </span> + <span id="busuanzi\_container\_site\_uv"> 来访 <span id="busuanzi\_value\_site\_uv"></span>人</span> + <span id="busuanzi\_container\_site\_pv">, 总访问 <span id="busuanzi\_value\_site\_pv"></span>次</span>  <% } %>  </p> |

修改主题后，建议先进行清理，不然可能由于缓存问题，导致本地预览与部署不一致，KO !

|  |  |
| --- | --- |
| 1 2 | hexo clean hexo d -g |

### 样式

#### 修改 logo 和 favicon

[用 Python 设计 Logo](https://www.alphalxy.com/2019/03/python-logo/)，并微调样式。

#### 按钮背景颜色增加渐变

diff:source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | .menu-list li ul  margin-right: 0 + .menu-list a + transition: background-color 0.3s ease-in-out  .menu-list a.level  display: flex |

#### card 增加浮动效果

:hover 时增大阴影，并增加动画属性 ease-in-out。

diff:source/css/style.styl

|  |  |
| --- | --- |
| 1 2 3 4 5 | .card  border-radius: 4px  box-shadow: 0 4px 10px rgba(0,0,0,0.05), 0 0 1px rgba(0,0,0,0.1) + &:hover + box-shadow: 0 6px 15px rgba(0,0,0,0.15), 0 0 1px rgba(0,0,0,0.1) |

diff:source/js/animation.js

|  |  |
| --- | --- |
| 1 2 3 | element.style.transform = ''; - element.style.transition = 'opacity 0.3s ease-out, transform 0.3s ease-out'; + element.style.transition = 'opacity 0.3s ease-out, transform 0.3s ease-out, box-shadow 0.3s ease-in-out'; |

#### 更新

2020-02-16 合并了 [2.7.0](https://github.com/ppoffice/hexo-theme-icarus/releases/tag/2.7.0) 版本，冲突不算太多，有两个点需要注意一下。

icarus 的配置里面，plugins.clipboard 需要移除，相关的配置移动到 article.highlight.clipboard。

\_config.yml

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | article:  # Code highlight settings  highlight:  # Code highlight themes  # https://github.com/highlightjs/highlight.js/tree/master/src/styles  theme: github-gist  # Show code copying button  clipboard: true  # Default folding status of the code blocks. Can be "", "folded", "unfolded"  fold: unfolded |

如果之前 links widget 没有配置链接，可能会遇到编译错误，需要增加一下判断。

diff:layout/widget/links.locals.js

|  |  |
| --- | --- |
| 1 2 3 4 5 | const links = get\_config\_from\_obj(locals.widget, 'links'); - if (Object.keys(links).length == 0) { + if (!links || Object.keys(links).length == 0) {  return null;  } |

#### 总结

这里只列举了部分改动，详细的差异可以查看 [diff](https://github.com/ppoffice/hexo-theme-icarus/compare/2.7.0...AlphaLxy:master)。

### 参考资料

[icarus文档](https://blog.zhangruipeng.me/hexo-theme-icarus/)

https://blog.zhangruipeng.me/hexo-theme-icarus/

[自定义 icarus](https://www.alphalxy.com/2019/03/customize-icarus/)

**https://www.alphalxy.com/2019/03/customize-icarus/**

### 修改要点

主题配置文件

Share button

89行注释掉

{!index ? <Share config={config} page={page} helper={helper} /> : null}

将捐赠注释掉后会报错 恢复

93行

\themes\icarus\layout\common\article.jsx"(93,17): {/\* Donate button \*/}

注释掉 {!index ? <Donates config={config} helper={helper} /> : null}

{/\* Comment \*/}

注释掉

{!index ? <Comment config={config} page={page} helper={helper} /> : null}

### 主题pinghsu

typecho 主题写了一个非常漂亮的 hexo 主题，地址是：https://github.com/cccyb/hexo-theme-pinghsu，欢迎大家使用。

#### 前言

本文将会从零开始编写一个简单的Hexo博客主题，目的是了解一个Hexo博客主题的构成以及如何编写，因此，本示例中的博客页面样式不做过多描绘，样式主要参考 Hexo theme 中的 Noise 主题。

在开始前，你需要对以下的一些知识点有必要的了解：

* 模板引擎语法
* CSS预处理器
* YML语法
* Hexo文档

本文使用的模板引擎为 ejs，使用的 CSS 预处理器为 stylus。这也是 hexo 项目预装了的 render 插件，如果想使用其他模板引擎或者其他 CSS 预处理器，可以安装相对应的 render 插件。

本文的代码：https://github.com/cccyb/theme-example

#### 目录结构

根据下图，在themes目录下新建一个theme-example文件夹作为我们的主题目录。

如图所示，一个hexo主题的目录主要包括以下五部分：

* languages：用于国际化的语言文件
* layout：主题布局模板文件
* scripts：hexo脚本插件目录，可以编写一些辅助函数脚本
* source：资源文件目录，包括页面样式，js脚本等
* \_config.yml：主题配置文件

#### 局部模板

我们通过分析常见的博客网站可以知道，大部分的博客网站都由三部分组成：顶部导航栏，中间内容区域，以及底部信息展示区域。每次点击导航栏选项跳转页面时，顶部导航栏以及底部信息展示区域是不变的，只是中间的内容区域重新渲染，因此，我们可以将通用的代码抽离成局部模板以复用。

我们在layout目录下新建\_partial目录，在该目录下添加head.ejs,header.ejs以及footer.ejs文件。

layout/\_partial/head.ejs:

<head>  
    <meta http-equiv="content-type" content="text/html; charset=utf-8">  
    <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0" name="viewport">  
    <title>标题</title>  
</head>

layout/\_partial/header.ejs:

<header>我是导航栏</header>

layout/\_partial/footer.ejs:

<footer>我是底部信息</footer>

我们在layout中创建layout.ejs，并引入head.ejs,header/ejs和footer.ejs文件，layout.ejs文件是通用的布局文件模板，我们在后面新增的ejs文件都会继承layout.ejs，并将其内容填充到body中。

layout/layout.ejs:

<!DOCTYPE html>  
<html>  
<%- partial('\_partial/head') %>  
<body>  
    <div class="container">  
    <%- partial('\_partial/header') %>  
    <%- body %>  
    <%- partial('\_partial/footer') %>  
    </div>  
</body>  
</html>

#### 首页

首页是我们的网站加载完毕后的第一个页面。

我们在 layout 中创建 index.ejs 文件，index.ejs首页将会继承layout.ejs布局模板生成 HTML 文件。

partial()函数的作用是可以引入其他模板文件，详情参考hexo文档

layout/index.ejs:

<h1>Hello World</h1>

修改**站点配置文件**中的主题配置，使用我们刚刚创建的 theme-example 主题：

# Extensions  
## Plugins: https://hexo.io/plugins/  
## Themes: https://hexo.io/themes/  
theme: theme-example

运行 hexo server 开启 Hexo 本地服务器预览，访问 http://localhost:4000/。

#### 编写导航栏和底部信息

前面，我们只是搭了个页面框架，接下来我们就将开始正式开始一步步的完善我们的主题，以下两个文档我们将频繁的使用，最好可以提前阅读一遍有个了解：

* Hexo | 变量
* Hexo | 辅助函数

layout/\_partial/head.ejs:

<head>  
    <meta http-equiv="content-type" content="text/html; charset=utf-8">  
    <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0" name="viewport">  
    <title><%= config.title %></title>  
</head>

这里我们使用了config全局变量，该变量包含的是站点配置（即站点根目录下 \_config.yml 中的配置）。除此之外，我还有将经常使用的是theme变量，该变量是主题配置（即主题根目录下 \_config.yml 中的配置），其他变量参见hexo文档。

编辑导航栏部分，layout/\_partial/header.ejs：

<header class="header">  
    <div class="title">  
        <a href="<%= url\_for() %>" class="logo"><%= config.title %></a>  
    </div>  
    <nav class="navbar">  
        <ul class="menu">  
            <li class="menu-item">  
                <a href="/" class="menu-item-link">Home</a>  
            </li>  
            <li class="menu-item">  
                <a href="/categories" class="menu-item-link">Categories</a>  
            </li>  
            <li class="menu-item">  
                <a href="/tags" class="menu-item-link">Tags</a>  
            </li>  
            <li class="menu-item">  
                <a href="/archives" class="menu-item-link">Archives</a>  
            </li>  
        </ul>  
    </nav>  
</header>

编辑底部信息部分，layout/\_partial/footer.ejs：

<footer>  
    <p>Theme is <a href="/" target="\_blank">Theme-example</a> by <a href="<%= config.url %>" target="\_blank"><%= config.author %></a></p>  
    <p>Powered by <a href="https://hexo.io/" target="\_blank" rel="nofollow">hexo</a> &copy; <%- date(Date.now(), 'YYYY') %></p>  
</footer>

这样，我们就得到了一个包含导航栏和底部信息的简单页面。

#### 添加主题配置

实际上我们需要让导航菜单根据我们的需要显示不同的项，上面这种写法不方便修改。所以我们会在主题的配置文件中添加导航菜单的配置。在 theme-example 下的配置文件 \_config.yml，在其中添加需要配置的字段。然后可以通过 theme 这个变量来拿到该配置文件中的配置。

theme-example/\_config.yml:

menu:  
  home: /  
  categories: /categories  
  tags: /tags  
  archives: /archives

这样我们就可以在 header.ejs 中使用 theme.menu 获取到导航菜单的设置。将 header.ejs 修改为：

<header class="header">  
    <div class="title">  
        <a href="<%= url\_for() %>" class="logo"><%= config.title %></a>  
    </div>  
    <nav class="navbar">  
        <ul class="menu">  
            <% for (name in theme.menu) { %>  
            <li class="menu-item">  
                <a href="<%- url\_for(theme.menu[name]) %>" class="menu-item-link"><%= name %></a>  
            </li>  
            <% } %>  
        </ul>  
    </nav>  
</header>

这样，我们就可以动态的配置导航的信息了，我们还可以在主题配置文件中添加其他配置项供我们使用。

#### 添加文章列表

接着我们完善首页的模板，使其能够显示文章列表。前面已经说过 Hexo 提供了各种有用的变量，在这里将会使用到 page 这个变量。page 会根据不同的页面拥有不同的属性。具体有什么属性，可以获取到哪些数据可以查看这里。

那么这里我们会使用 page 变量的 posts 属性拿到文章数据的集合。编辑 index.ejs 文件：

<section class="posts">  
    <% page.posts.each(function (post) { %>  
    <article class="post">  
        <div class="post-title">  
            <a class="post-title-link" href="<%- url\_for(post.path) %>"><%= post.title %></a>  
        </div>  
        <div class="post-content">  
            <%- post.content %>  
        </div>  
        <div class="post-meta">  
            <span class="post-time"><%- date(post.date, "YYYY-MM-DD") %></span>  
        </div>  
    </article>  
    <% }) %>  
</section>

从 page.posts 中获取单篇文章的数据，并获取文章的标题，内容等数据填充到模板中。处理文章创建时间的时候使用了 date() 函数，这是 Hexo 提供的时间处理的辅助函数。

由于首页显示文章内容时使用的是 post.content，即文章的全部内容。所以首页会显示每一篇文章的内容，实际上我们并不想在首页显示那么多内容，只想显示文章的摘录。

Hexo 提供了 excerpt 属性来获取文章的摘录部分，不过这里需要在文章中添加一个 <!--more--> 标记。添加了这个标记之后，post.excerpt 将会获取到标记之前的内容。如果没有这个标记，那么 post.excerpt 会是空的。所以我们可以把首页文章内容部分的 post.content 替换成 post.excerpt。

<div class="post-content">  
    <%- post.excerpt %>  
</div>

#### 添加样式

到目前为止，我们完成了首页的整体页面结构，不过由于没添加css样式，因此整体页面非常难看，所以我们需要给页面加上一些样式来美化一下我们的页面。

由于 Hexo 在新建项目的时候会安装 hexo-renderer-stylus 这个插件，所以我们无需其他步骤，只需要将样式文件放到 css 文件夹中。Hexo 在生成页面的时候会将 source 中的所有文件复制到生成的 public 文件中，并且在此之前会编译 styl 为 css 文件。

在 css 文件夹中创建 style.styl，编写一些基础的样式，并把所有样式 import 到这个文件。所以最终编译之后只会有 style.css 一个文件。创建 \_partial/header.styl 与 \_partial/post.styl 存放页面导航以及文章的样式，并且在 style.styl 中 import 这两个文件。

\_partial/header.styl：

.header {  
    margin-top: 2em  
    display: flex  
    align-items: baseline  
    justify-content: space-between  
  
    .blog-title .logo {  
        color: #AAA;  
        font-size: 2em;  
        font-family: "Comic Sans MS",cursive,LiSu,sans-serif;  
        text-decoration: none;  
    }  
  
    .menu {  
        margin: 0;  
        padding: 0;  
  
        .menu-item {  
            display: inline-block;  
            margin-right: 10px;  
        }  
  
        .menu-item-link {  
            color: #AAA;  
            text-decoration: none;  
  
            &:hover {  
                color: #368CCB;  
            }  
        }  
    }  
}

index.styl:

.post {  
    margin: 1em auto;  
    padding: 30px 50px;  
    background-color: #fff;  
    border: 1px solid #ddd;  
    box-shadow: 0 0 2px #ddd;  
}  
  
.posts  {  
    .post:first-child {  
        margin-top: 0;  
    }  
  
    .post-title {  
        font-size: 1.5em;  
  
        .post-title-link {  
            color: #368CCB;  
            text-decoration: none;  
        }  
    }  
  
    .post-content {  
        a {  
            color: #368CCB;  
            text-decoration: none;  
        }  
    }  
  
    .post-meta {  
        color: #BABABA;  
    }  
}

style.styl:

body {  
    background-color: #F2F2F2;  
    font-size: 1.25rem;  
    line-height: 1.5;  
}  
  
.container {  
    max-width: 960px;  
    margin: 0 auto;  
}  
  
@import "\_partial/header";  
@import "\_partial/index";

最后，我们需要把样式添加到页面中，这里使用了另外一个辅助函数 css():

layout/\_partial/head.ejs:

<head>  
    <meta http-equiv="content-type" content="text/html; charset=utf-8">  
    <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0" name="viewport">  
    <title><%= config.title %></title>  
    <%- css('css/style.styl') %>  
</head>

至此，我们会看到站点的首页是这个样子的：

#### 添加分页

在站点的 source/\_post/ 目录下存放的是我们的文章，现在我们把原本的 hello-world.md 复制黏贴 10+ 次，再查看站点首页。会发现，首页只显示了 10 篇文章。

首页显示的文章数量我们可以通过站点配置文件中的 per\_page 字段来修改，但是我们不可能把所有文章都放在一页，所以我们现在来添加文章列表的分页。

新建 \_partial/paginator.ejs:

<% if (page.total > 1){ %>  
    <nav class="page-nav">  
    <%- paginator({  
        prev\_text: "&laquo; Prev",  
        next\_text: "Next &raquo;"  
    }) %>  
    </nav>  
<% } %>

在 index.ejs 中添加这个文件的内容：

...  
</section>  
<%- partial('\_partial/paginator') %>

这里我们使用到了另外的一个辅助函数 paginator，它能够帮助我们插入分页链接，这里我们只是实现了一个最基本的分页，具体的样式可以自行添加，或者根据文档使用其他配置自定义分页。

#### 添加文章详情页

文章详情页对应的布局文件是 post.ejs，新建 post.ejs:

<article class="post">  
    <div class="post-title">  
        <h2 class="title"><%= page.title %></h2>  
    </div>  
    <div class="post-meta">  
        <span class="post-time"><%- date(page.date, "YYYY-MM-DD") %></span>  
    </div>  
    <div class="post-content">  
        <%- page.content %>  
    </div>  
</article>

由于这里是文章的模板，所以变量 page 表示的是文章的数据，而不是首页的文章数据集合。

#### 添加归档页

创建归档页使用的模板文件 archive.ejs:

<section class="archive">  
    <ul class="post-archive">  
    <% page.posts.each(function (post) { %>  
        <li class="post-item">  
            <span class="post-date"><%= date(post.date, "YYYY-MM-DD") %></span>  
            <a class="post-title" href="<%- url\_for(post.path) %>"><%= post.title %></a>  
        </li>  
    <% }) %>  
    </ul>  
</section>  
<%- partial('\_partial/paginator') %>

其实结构跟首页差不多，只是不显示文章内容而已。添加归档页的样式：

css/\_partial/archive.styl:

.archive {  
    margin: 1em auto;  
    padding: 30px 50px;  
    background-color: #fff;  
    border: 1px solid #ddd;  
    box-shadow: 0 0 2px #ddd;  
  
    .post-archive {  
        list-style: none;  
        padding: 0;  
  
        .post-item {  
            margin: 5px 0;  
  
            .post-date {  
                display: inline-block;  
                margin-right: 10px;  
                color: #BABABA;  
            }  
  
            .post-title {  
                color: #368CCB;  
                text-decoration: none;  
            }  
        }  
    }  
}

#### 添加分类页

分类页跟归档页也类似，相当于根据不同的分类进行归档。

分类页和标签页的模板编写比较特殊，本质上，分类页和标签页属于自定义页面，我们需要新建自定义页面模板page.ejs:

<% if (is\_current(theme.menu.categories)) { %>  
<%- partial('\_partial/category') %>  
<% } else if (is\_current(theme.menu.tags)) { %>  
<%- partial('\_partial/tag') %>  
<% } else { %>  
<%- partial('\_partial/custom') %>  
<% } %>

这里，我们需要根据当前自定义页面的类型来决定渲染何种自定义页面模板。

新建\_partial/category.ejs:

<section class="archive">  
    <ul class="post-archive">  
    <% site.categories.each(function (category) { %>  
        <span><%= category.name %></span>  
        <% category.posts.forEach(function(post) { %>  
        <li class="post-item">  
            <span class="post-date"><%= date(post.date, "YYYY-MM-DD") %></span>  
            <a class="post-title" href="<%- url\_for(post.path) %>"><%= post.title %></a>  
        </li>  
        <% }) %>  
    <% }) %>  
    </ul>  
</section>

site.categories包括了站点所有的分类信息，可以遍历获取分类信息，其中category.posts又包含了该分类的所有文章信息。

需要注意的是，要想在页面中展示分类页，需要先执行hexo new page categories生成分类页面，并添加type为categories：

title: categories  
date: 2019-02-25 18:19:55  
type: "categories"  
---

后续的标签页类似。

#### 添加标签页

标签页与分类页及其类似，只是根据标签进行归档。

新建\_partial/tag.ejs:

<section class="archive">  
    <ul class="post-archive">  
    <% site.tags.each(function (tag) { %>  
        <span><%= tag.name %></span>  
        <% tag.posts.forEach(function(post) { %>  
        <li class="post-item">  
            <span class="post-date"><%= date(post.date, "YYYY-MM-DD") %></span>  
            <a class="post-title" href="<%- url\_for(post.path) %>"><%= post.title %></a>  
        </li>  
        <% }) %>  
    <% }) %>  
    </ul>  
</section>

#### 添加自定义页面

自定义页面与文章详情页类似。

新建\_partial/custom.ejs:

<article class="post">  
    <div class="post-title">  
        <h2 class="title"><%= page.title %></h2>  
    </div>  
    <div class="post-meta">  
        <span class="post-time"><%- date(page.date, "YYYY-MM-DD") %></span>  
    </div>  
    <div class="post-content">  
        <%- page.content %>  
    </div>  
</article>

我们在主题配置文件中添加自定义页面的菜单：

menu:  
  home: /  
  categories: /categories  
  tags: /tags  
  archives: /archivesz  
  about: /about

执行hexo new page about进行手动生成页面，编辑文件内容即可。

参考https://github.com/Shen-Yu/hexo-theme-ayer

## 主题：butterfly

### Installation

Stable branch [recommend]:

git clone -b master https://github.com/jerryc127/hexo-theme-butterfly.git themes/butterfly

Dev branch:

git clone -b dev https://github.com/jerryc127/hexo-theme-butterfly.git themes/butterfly

### Configuration

Set theme in the hexo work folder's root config file \_config.yml:

theme: butterfly

If you don't have pug & stylus renderer, try this:

npm install hexo-renderer-pug hexo-renderer-stylus

### Documentation

Find in [Butterfly Docs](https://demo.jerryc.me/posts/21cfbf15/). it supports zh-TW and zh-CN

You can also join the Chat Group.

[💬 Join QQ Group [recommend]](https://jq.qq.com/?_wv=1027&k=5KI3Jqb)

[💬 Join Telegram Group](https://t.me/hexo_butterfly)

### Screenshots

[](https://user-images.githubusercontent.com/16351105/58887365-1272f780-8718-11e9-9329-3292c6ba20d4.png)

[](https://user-images.githubusercontent.com/16351105/58887457-3cc4b500-8718-11e9-9417-2bdea603c92e.png)

[](https://user-images.githubusercontent.com/16351105/69338594-7d03f980-0c9e-11ea-8b64-7f165e6508e2.png)

### About

🦋 A Hexo Theme: Butterfly

#### Topics

[hexo](https://github.com/topics/hexo) [hexo-theme](https://github.com/topics/hexo-theme) [theme](https://github.com/topics/theme)

#### Resources

[Readme](https://github.com/jerryc127/hexo-theme-butterfly#readme)

#### License

[MIT License](https://github.com/jerryc127/hexo-theme-butterfly/blob/dev/LICENSE)

### [Releases 15](https://github.com/jerryc127/hexo-theme-butterfly/releases)

[3.0.0 RC 1Latest](https://github.com/jerryc127/hexo-theme-butterfly/releases/tag/3.0.0-rc1)

[on 16 Jun](https://github.com/jerryc127/hexo-theme-butterfly/releases/tag/3.0.0-rc1)

[+ 14 releases](https://github.com/jerryc127/hexo-theme-butterfly/releases)

### [Contributors 11](https://github.com/jerryc127/hexo-theme-butterfly/graphs/contributors)

* [](https://github.com/jerryc127)
* [](https://github.com/eallion)
* [](https://github.com/noraj)
* [](https://github.com/larscheng)
* [](https://github.com/Srkeo)
* [](https://github.com/flymin)
* [](https://github.com/sviptzk)
* [](https://github.com/bmyjacks)
* [](https://github.com/theWorkFsz)
* [](https://github.com/HolgerHuo)
* [](https://github.com/SerokSSR)

### Languages

* [CSS41.8%](https://github.com/jerryc127/hexo-theme-butterfly/search?l=css)
* [HTML32.4%](https://github.com/jerryc127/hexo-theme-butterfly/search?l=html)
* [JavaScript25.8%](https://github.com/jerryc127/hexo-theme-butterfly/search?l=javascript)
* © 2020 GitHub, Inc.
* language

hexo-theme-butterfly 是基於 Molunerfinn 的 hexo-theme-melody 的基礎上進行開發的。

文檔也是在 hexo-theme-melody 的文檔基礎上修改。因為一些配置變更導致與原主題配置上有部分區別。故如果安裝 hexo-theme-butterfly 主題，請參考這篇文檔。

安裝

穩定版【建議】

在你的博客根目錄裏

Powershell

1

git clone -b master https://github.com/jerryc127/hexo-theme-butterfly.git themes/butterfly

測試版

測試版可能存在 Bugs，追求穩定的請安裝穩定版

如果想要安裝比較新的 dev 分支，可以

Powershell

1

git clone -b dev https://github.com/jerryc127/hexo-theme-butterfly.git themes/butterfly

應用主題

修改站點配置文件\_config.yml，把主題改為 butterfly

Yaml

1

theme: butterfly

安裝插件

如果你沒有 pug 以及 stylus 的渲染器，請下載安裝：

Powershell

1

npm install hexo-renderer-pug hexo-renderer-stylus --save

升級建議

為了減少升級主題後帶來的不便，Butterfly 使用了 data files 特性。

推薦把主題默認的配置文件\_config.yml 複製到 Hexo 根目錄下的 source/\_data/ 目錄下，然後將文件名改為 butterfly.yml（如果 source/\_data/ 的目錄不存在就創建一個）。

升級方法： 在主題目錄下進行 git pull

注意，如果你創建了 butterfly.yml, 它將會替換主題默認配置文件\_config.yml 裏的配置項 (不是合併而是替換)

採用 butterfly.yml 的目的是，因爲升級主題的時候，有可能會覆蓋到配置文件，以至於每次更新的時候都需要重新配置文件。如果使用 butterfly.yml，就算主題目錄下的\_config.yml 被覆蓋，主題只會去 butterfly.yml 讀取配置。

由於主題在添加功能或者修復 Bugs 的情況下，可能會涉及到配置文件的修改。這時候，如果升級主題，需要把新增加的配置添加到 butterfly.yml 去，不然很大機會會出現報錯。

Front-matter

Front-matter 是檔案最上方以 --- 分隔的區域，用於指定個別檔案的變數。

如果標注可選的參數，可根據自己需要添加，不用全部都寫在 markdown 裏

Page Front-matter

Markdown

---

title:

date:

updated:

type:

comments:

description:

keywords:

top\_img:

mathjax:

katex:

aside:

aplayer：

highlight\_shrink：

---

寫法 解釋

title 【必需】頁面標題

date 【必需】頁面創建日期

type 【必需】標籤、分類和友情鏈接三個頁面需要配置

updated 【可選】頁面更新日期

description 【可選】頁面描述

keywords 【可選】頁面關鍵字

comments 【可選】顯示頁面評論模塊 (默認 true)

top\_img 【可選】頁面頂部圖片

mathjax 【可選】顯示 mathjax (當設置 mathjax 的 per\_page: false 時，才需要配置，默認 false)

katex 【可選】顯示 katex (當設置 katex 的 per\_page: false 時，才需要配置，默認 false)

aside 【可選】顯示側邊欄 (默認 true)

aplayer 【可選】在需要的頁面加載 aplayer 的 js 和 css, 請參考文章下面的音樂 配置

highlight\_shrink 【可選】配置代碼框是否展開 (true/false)(默認為設置中 highlight\_shrink 的配置)

Post Front-matter

Markdown

---

title:

date:

updated:

tags:

categories:

keywords:

description:

top\_img:

comments：

cover:

toc:

toc\_number:

auto\_open:

copyright:

mathjax:

katex:

aplayer：

highlight\_shrink：

---

寫法 解釋

title 【必需】文章標題

date 【必需】文章創建日期

updated 【可選】文章更新日期

tags 【可選】文章標籤

categories 【可選】文章分類

keywords 【可選】文章關鍵字

description 【可選】文章描述

top\_img 【可選】文章頂部圖片

cover 【可選】文章縮略圖 (如果沒有設置 top\_img, 文章頁頂部將顯示縮略圖，可設為 false / 圖片地址 / 留空)

comments 【可選】顯示文章評論模塊 (默認 true)

toc 【可選】顯示文章 TOC (默認為設置中 toc 的 enable 配置)

toc\_number 【可選】顯示 toc\_number (默認為設置中 toc 的 number 配置)

auto\_open 【可選】是否自動打開 TOC (默認為設置中 toc 的 auto\_open 配置)

copyright 【可選】顯示文章版權模塊 (默認為設置中 post\_copyright 的 enable 配置)

mathjax 【可選】顯示 mathjax (當設置 mathjax 的 per\_page: false 時，才需要配置，默認 false)

katex 【可選】顯示 katex (當設置 katex 的 per\_page: false 時，才需要配置，默認 false)

aplayer 【可選】在需要的頁面加載 aplayer 的 js 和 css, 請參考文章下面的音樂 配置

highlight\_shrink 【可選】配置代碼框是否展開 (true/false)(默認為設置中 highlight\_shrink 的配置)

### 標籤頁

前往你的 Hexo 博客的根目錄

輸入 hexo new page tags

你會找到 source/tags/index.md 這個文件

修改這個文件：

Markdown

---

title: 標籤

date: 2018-01-05 00:00:00

type: "tags"

---

### 分類頁

前往你的 Hexo 博客的根目錄

輸入 hexo new page categories

你會找到 source/categories/index.md 這個文件

修改這個文件：

Markdown

---

title: 分類

date: 2018-01-05 00:00:00

type: "categories"

---

### 友情鏈接

為你的博客創建一個友情鏈接！

創建友情鏈接頁面

前往你的 Hexo 博客的根目錄

輸入 hexo new page link

你會找到 source/link/index.md 這個文件

修改這個文件：

Markdown

---

title: 友情鏈接

date: 2018-06-07 22:17:49

type: "link"

---

友情鏈接添加

在 Hexo 博客目錄中的 source/\_data，創建一個文件 link.yml

Yml

- class\_name: 友情鏈接

class\_desc: 那些人，那些事

link\_list:

- name: JerryC

link: https://jerryc.me/

avatar: https://jerryc.me/image/avatar.png

descr: 今日事,今日畢

- name: Hexo

link: https://hexo.io/zh-tw/

avatar: https://d33wubrfki0l68.cloudfront.net/6657ba50e702d84afb32fe846bed54fba1a77add/827ae/logo.svg

descr: 快速、簡單且強大的網誌框架

- class\_name: 網站

class\_desc: 值得推薦的網站

link\_list:

- name: Youtube

link: https://www.youtube.com/

avatar: https://i.loli.net/2020/05/14/9ZkGg8v3azHJfM1.png

descr: 視頻網站

- name: Weibo

link: https://www.weibo.com/

avatar: https://i.loli.net/2020/05/14/TLJBum386vcnI1P.png

descr: 中國最大社交分享平台

- name: Twitter

link: https://twitter.com/

avatar: https://i.loli.net/2020/05/14/5VyHPQqR6LWF39a.png

descr: 社交分享平台

class\_name 和 class\_desc 支持 html 格式書寫，如不需要，也可以留空。

友情鏈接界面設置

由 2.2.0 起，友情鏈接界面可以由用户自己自定義，只需要在友情鏈接的 md 檔設置就行，以普通的 Markdown 格式書寫。

### 音樂

音樂界面使用了插件 hexo-tag-aplayer。

使用方法請參考插件的文檔。

音樂頁面只是普通的 page 頁，按普通頁面操作生成就行。

以下內容可供選擇配置

注意： 仍需要安裝插件 hexo-tag-aplayer

插件會在每一個文件都插入 js 和 css，為了避免這一情況，3.0 版本內置了 aplayer 需要的 css 和 js。

首先在 Hexo 根目錄\_config 裏配置 asset\_inject 為 false

Yaml

1

2

aplayer:

asset\_inject: false

然後在你需要使用 aplayer 的頁面 Front-matter 添加

Markdown

aplayer: true

這樣只會在需要 aplayer 的頁面插入 js 和 css。

電影

電影界面使用了插件 hexo-douban。

使用方法請參考插件的文檔。

注意：hexo-douban 會主動生成頁面，所以不需要自己創建。對應網頁的 top\_img 可以在 butterfly.yml 修改。

### 404 頁面

主題內置了一個簡單的 404 頁面，可在設置中開啟

本地預覽時，訪問出錯的網站是不會跳到 404 頁面的。

如需本地預覽，請訪問 http://localhost:4000/404.html

Yaml

1

2

3

4

5

# A simple 404 page

error\_404:

enable: true

subtitle: "頁面沒有找到"

background:

配置文件説明

站點配置文件\_config.yml 是 hexo 工作目錄下的配置文件

butterfly.yml（如有創建）是 Butterfly 的配置文件，等同於主題目錄下的\_config.yml

語言

修改站點配置文件 \_config.yml

默認語言是 en

主題支持三種語言

default(en)

zh-CN (簡體中文)

zh-TW (繁體中文)

網站資料

修改網站各種資料，例如標題、副標題和郵箱等個人資料，請修改博客根目錄的\_config.yml

### 导航菜单

配置 butterfly.yml

Yaml

menu:

Home: / || fas fa-home

Archives: /archives/ || fas fa-archive

Tags: /tags/ || fas fa-tags

Categories: /categories/ || fas fa-folder-open

List||fas fa-list:

- Music || /music/ || fas fa-music

- Movie || /movies/ || fas fa-video

Link: /link/ || fas fa-link

About: /about/ || fas fa-heart

必須是 /xxx/，後面 || 分開，然後寫圖標名。

注意： 導航的文字可自行更改

例如：

Markdown

menu:

首頁: / || fas fa-home

時間軸: /archives/ || fas fa-archive

標籤: /tags/ || fas fa-tags

分類: /categories/ || fas fa-folder-open

清單||fa fa-heartbeat:

- 音樂 || /music/ || fas fa-music

- 照片 || /Gallery/ || fas fa-images

- 電影 || /movies/ || fas fa-video

友鏈: /link/ || fas fa-link

關於: /about/ || fas fa-heart

### 代碼

代碼塊中的所有功能只適用於 Hexo 默認的 highlight 渲染

如果使用第三方的渲染器，不一定會有效

代碼高亮主題

默認主題

自定義主題

Butterfly 支持 6 種代碼高亮樣式：

default

darker

pale night

light

ocean

mac

配置 butterfly.yml

Yaml

1

highlight\_theme: light

default

darker

pale night

light

ocean

mac

代碼複製

主題支持代碼複製功能

配置 butterfly.yml

Yaml

1

highlight\_copy: true

代碼框展開 / 關閉

在默認情況下，代碼框自動展開，可設置是否所有代碼框都關閉狀態，點擊 > 可展開代碼

true 全部代碼框不展開，需點擊 > 打開

false 代碼狂展開，有 > 點擊按鈕

none 不顯示 > 按鈕

配置 butterfly.yml

Yaml

1

highlight\_shrink: true #代碼框不展開，需點擊 '>' 打開

你也可以在 post/page 頁對應的 markdown 文件 front-matter 添加 highlight\_shrink 來獨立配置。

當 butterfly.yml 中的 highlight\_shrink 設為 true 時，可在 front-matter 添加 highlight\_shrink: false 來單獨配置文章展開代碼框。

當 butterfly.yml 中的 highlight\_shrink 設為 false 時，可在 front-matter 添加 highlight\_shrink: true 來單獨配置文章收縮代碼框。

highlight\_shrink: true

highlight\_shrink: false

highlight\_shrink: none

### 代碼換行

在默認情況下，hexo-highlight 在編譯的時候不會實現代碼自動換行。如果你不希望在代碼塊的區域裏有橫向滾動條的話，那麼你可以考慮開啟這個功能。

配置 butterfly.yml

Yaml

1

code\_word\_wrap: true

然後找到你站點的 Hexo 配置文件\_config.yml，將 line\_number 改成 false:

Yaml

highlight:

enable: true

line\_number: false

auto\_detect: false

tab\_replace:

設置 code\_word\_wrap 之前:

設置 code\_word\_wrap 之後:

### 社交圖標

Butterfly 支持 font-awesome v5 圖標.

書寫格式 圖標名：url || 描述性文字

Yaml

social:

fab fa-github: https://github.com/xxxxx || Github

fas fa-envelope: mailto:xxxxxx@gmail.com || Email

圖標名可在這尋找

PC:

Mobile:

主頁文章節選 (自動節選和文章頁 description)

因為主題 UI 的關係，主頁文章節選只支持自動節選和文章頁description。

在 butterfly 裏，有三種可供選擇

description 只顯示 description

both 優先選擇 description，如果沒有配置 description，則顯示自動節選的內容

auto\_excerpt 只顯示自動節選

配置 butterfly.yml

Yaml

1

2

3

index\_post\_content:

method: 3

length: 500 # if you set method to 2 or 3, the length need to config

description 在 front-matter 裏添加

### 頂部圖

頂部圖有 2 種配置：具體 url 和（留空，true 和 false，三個效果一樣）

page 頁

當具體 url 時

主頁的頂部圖可以在 Butterfly.yml 設置 index\_img

archives 頁的頂部圖可以在 Butterfly.yml 設置 archive\_img

其他 page 頁的頂部圖可以在各自的 md 頁面設置 front-matter 中的 top\_img

頁面如果沒有設置各自的 top\_img，則會顯示 default\_top\_img 圖片

當頂部圖留空，true 和 false

主頁會顯示純顏色的頂部圖

其他 page 的頂部圖沒有設置時，也會顯示純顏色的頂部圖

post 頁

post 頁的頂部圖會優先顯示各自 front-matter 中的 top\_img, 如果沒有設置，則會縮略圖（即各自 front-matter 中的 cover)，如果沒有則會顯示顯示 default\_top\_img 圖片

### 文章置顶

要為文章置頂，你需要安裝插件 (hexo-generator-index-pin-top 或者 hexo-generator-indexed)

如果使用 hexo-generator-index-pin-top, 需要先卸載掉 hexo-generator-index，然後在文章的 front-matter 區域裏添加 top: true 屬性來把這篇文章置頂

如果使用 hexo-generator-indexed, 需要先卸載掉 hexo-generator-index，然後在文章的 front-matter 區域裏添加 sticky: 1 屬性來把這篇文章置頂。數值越大，置頂的優先級越大

### 文章封面

文章的 markdown 文檔上，在 Front-matter 添加 cover, 並填上要顯示的圖片地址。

如果不配置 cover, 可以設置顯示默認的 cover.

如果不想在首頁顯示 cover, 可以設置為 false

配置 butterfly.yml

Yaml

cover:

# 是否顯示文章封面

index\_enable: true

aside\_enable: true

archives\_enable: true

# 封面顯示的位置

# 三個值可配置 left , right , both

position: both

# 當沒有設置cover時，默認的封面顯示

default\_cover:

當配置多張圖片時，會隨機選擇一張作為 cover. 此時寫法應為

Yaml

1

2

3

4

default\_cover:

- https://cdn.jsdelivr.net/gh/jerryc127/CDN@latest/cover/default\_bg.png

- https://cdn.jsdelivr.net/gh/jerryc127/CDN@latest/cover/default\_bg2.png

- https://cdn.jsdelivr.net/gh/jerryc127/CDN@latest/cover/default\_bg3.png

left

right

both

### 文章頁相關配置

文章 meta 顯示

這個選項是用來顯示文章的相關信息的。

配置 butterfly.yml

Yaml

post\_meta:

page:

date\_type: both # created or updated or both 主頁文章日期是創建日或者更新日或都顯示

categories: true # true or false 主頁是否顯示分類

tags: true # true or false 主頁是否顯示標籤

post:

date\_type: both # created or updated or both 文章頁日期是創建日或者更新日或都顯示

categories: true # true or false 文章頁是否顯示分類

tags: true # true or false 文章頁是否顯示標籤

在文章頂部的資料，

date\_type: 可設置文章日期顯示創建日期 (created) 或者更新日期 (updated) 或者兩種都顯示 (both)

categories 是否顯示分類

tags 是否顯示標籤

文章版權

為你的博客文章展示文章版權和許可協議。

配置 butterfly.yml

Yaml

post\_copyright:

enable: true

decode: false

license: CC BY-NC-SA 4.0

license\_url: https://creativecommons.org/licenses/by-nc-sa/4.0/

由於 Hexo 4.1 開始，默認對網址進行解碼，以至於如果是中文網址，會被解碼，可設置 decode: true 來顯示中文網址。

如果有文章（例如：轉載文章）不需要顯示版權，可以在文章 Front-matter 單獨設置

Yaml

1

copyright: false

版權顯示截圖

### 文章打賞

在你每篇文章的結尾，可以添加打賞按鈕。相關二維碼可以自行配置。

對於沒有提供二維碼的，可配置一張軟件的 icon 圖片，然後在 link 上添加相應的打賞鏈接。用户點擊圖片就會跳轉到鏈接去。

link 可以不寫，會默認為圖片的鏈接。

配置 butterfly.yml

Yaml

reward:

enable: true

QR\_code:

- img: /image/wechat.jpg

link:

text: 微信

- img: /image/alipay.jpg

link:

text: 支付寶

### 文章隱藏

2.3.0 開始主題不再默認提供文章隱藏功能

如需要文章隱藏功能，請裝插件 hexo-generator-indexed 或者 hexo-hide-posts

TOC

在文章頁，會有一個目錄，用於顯示 TOC。

enable: 是否顯示 TOC

number: 是否顯示章節數

auto\_open: 可選擇進入文章頁面時，是否自動打開 sidebar 顯示 TOC

配置 butterfly.yml

Yaml

1

2

3

4

toc:

enable: true

number: true

auto\_open: true # auto open the sidebar

Demo

Toc PC

Toc Mobile

number: true

number: false

### 為特定的文章配置

在你的文章 md 文件的頭部，加入 toc\_number、toc 和 auto\_open，並配置 true 或者 false 即可。

主題會優先判斷文章 Markdown 的 Front-matter 是否有配置，如有，則以 Front-matter 的配置為準。否則，以 butterfly.yml 的配置為準

相關文章

相關文章推薦的原理是根據文章 tags 的比重來推薦

配置 butterfly.yml

Yaml

1

2

3

4

related\_post:

enable: true

limit: 6 # 顯示推薦文章數目

date\_type: created # or created or updated 文章日期顯示創建日或者更新日

### 文章錨點

開啟文章錨點後，當你在文章頁進行滾動時，文章鏈接會根據標題 ID 進行替換

(注意：每替換一次，會留下一個歷史記錄。所以如果一篇文章有很多錨點的話，網頁的歷史記錄會很多。)

配置 butterfly.yml

Yaml

1

2

3

# anchor

# when you scroll in post , the url will update according to header id.

anchor: ture

頭像

配置 butterfly.yml

Yaml

1

2

3

avatar:

img: /img/avatar.png

effect: true # 頭像會一直轉圈

圖片描述

可開啟圖片 Figcaption 描述文字顯示

配置 butterfly.yml

Yaml

1

photofigcaption: true

Footer 設置

博客年份

since 是一個來展示你站點起始時間的選項。它位於頁面的最底部。

配置 butterfly.yml

Yaml

1

since: 2018

頁腳自定義文本

footer\_custom\_text 是一個給你用來在頁腳自定義文本的選項。通常你可以在這裏寫聲明文本等。支持 HTML。

配置 butterfly.yml

Yaml

1

footer\_custom\_text: Hi, welcome to my <a href="https://jerryc.me/">blog</a>!

ICP

對於部分有備案的域名，可以在 ICP 配置顯示。

配置 butterfly.yml

Yaml

ICP:

enable: true

url: http://www.beian.miit.gov.cn/state/outPortal/loginPortal.action

text: 粵ICP備xxxx

icon: /img/icp.png

右下角按鈕

簡繁轉換

簡體繁體互換

右下角會有簡繁轉換按鈕。

配置 butterfly.yml

Yaml

translate:

enable: true

# 默認按鈕顯示文字(網站是簡體，應設置為'default: 繁')

default: 簡

#網站默認語言，1: 繁體中文, 2: 簡體中文

defaultEncoding: 1

#延遲時間,若不在前, 要設定延遲翻譯時間, 如100表示100ms,默認為0

translateDelay: 0

#當文字是簡體時，按鈕顯示的文字

msgToTraditionalChinese: "繁"

#當文字是繁體時，按鈕顯示的文字

msgToSimplifiedChinese: "簡"

簡體

繁體

### 夜間模式

右下角會有夜間模式按鈕

配置 butterfly.yml

Yaml

# dark mode

darkmode:

enable: true

# dark mode和 light mode切換按鈕

button: true

autoChangeMode: false

V2.0.0 開始增加一個選項，可開啟自動切換 light mode 和 dark mode

autoChangeMode: 1 跟隨系統而變化，不支持的瀏覽器 / 系統將按照時間晚上 6 點到早上 6 點之間切換為 dark mode

autoChangeMode: 2 只按照時間 晚上 6 點到早上 6 點之間切換為 dark mode, 其餘時間為 light mode

autoChangeMode: false 取消自動切換

### 閲讀模式

閲讀模式下會去掉除文章外的內容，避免幹擾閲讀。

只會出現在文章頁面，右下角會有閲讀模式按鈕。

配置 butterfly.yml

Yaml

1

readmode: true

側邊欄設置

側邊排版

可自行決定哪個項目需要顯示，可決定位置，也可以設置不顯示側邊欄。

配置 butterfly.yml

Yaml

aside:

enable: true

mobile: true # 手機頁面（ 顯示寬度 < 768px ）是否顯示aside內容

position: right # left or right

card\_author:

enable: true

description:

card\_announcement:

enable: true

content: This is my Blog

card\_recent\_post:

enable: true

limit: 5 # if set 0 will show all

card\_categories:

enable: true

limit: 8 # if set 0 will show all

expand: none # none/true/false

card\_tags:

enable: true

limit: 40 # if set 0 will show all

color: false

card\_archives:

enable: true

type: monthly # yearly or monthly

format: MMMM YYYY # eg: YYYY年MM月

order: -1 # Sort of order. 1, asc for ascending; -1, desc for descending

limit: 8 # if set 0 will show all

card\_webinfo: true

position: left

position: right

card\_tags color: false

card\_tags color: true

### 訪問人數 busuanzi (UV 和 PV)

訪問 busuanzi 的官方網站查看更多的介紹。

配置 butterfly.yml

Yaml

busuanzi:

site\_uv: true

site\_pv: true

page\_pv: true

### 運行時間

網頁已運行時間

配置 butterfly.yml

Yaml

runtimeshow:

enable: true

publish\_date: 6/7/2018 00:00:00

##網頁開通時間

#格式: 月/日/年 時間

#也可以寫成 年/月/日 時間

標籤外掛（Tag Plugins）

標籤外掛是 Hexo 獨有的功能，並不是標準的 Markdown 格式。

以下的寫法，只適用於 Butterfly 主題，用在其它主題上不會有效果，甚至可能會報錯。使用前請留意

標籤外掛雖然能為主題帶來一些額外的功能和 UI 方面的強化，但是，標籤外掛也有明顯的限制，使用時請留意。

### Note (Bootstrap Callout)

移植於 next 主題

配置 butterfly.yml

Yaml

note:

# Note tag style values:

# - simple bs-callout old alert style. Default.

# - modern bs-callout new (v2-v3) alert style.

# - flat flat callout style with background, like on Mozilla or StackOverflow.

# - disabled disable all CSS styles import of note tag.

style: simple

icons: false

border\_radius: 3

# Offset lighter of background in % for modern and flat styles (modern: -12 | 12; flat: -18 | 6).

# Offset also applied to label tag variables. This option can work with disabled note tag.

light\_bg\_offset: 0

用法

Markdown

1

2

3

4

5

6

7

8

{% note [class] [no-icon] %}

Any content (support inline tags too.io).

{% endnote %}

[class] : default | primary | success | info | warning | danger.

[no-icon] : Disable icon in note.

All parameters are optional.

例如：

Markdown

{% note default %}

default 提示塊標籤

{% endnote %}

{% note primary no-icon %}

primary 提示塊標籤

{% endnote %}

{% note success %}

success 提示塊標籤

{% endnote %}

{% note info %}

info 提示塊標籤

{% endnote %}

{% note warning %}

warning 提示塊標籤

{% endnote %}

{% note danger %}

danger 提示塊標籤

{% endnote %}

style: simple

style: modern

style: flat

style: disabled

Gallery 相冊圖庫

2.2.0 以上提供

一個圖庫集合。

寫法

Code

1

2

3

4

5

<div class="gallery-group-main">

{% galleryGroup name description link img-url %}

{% galleryGroup name description link img-url %}

{% galleryGroup name description link img-url %}

</div>

name：圖庫名字

description：圖庫描述

link：連接到對應相冊的地址

img-url：圖庫封面的地址

例如：

Code

1

2

3

4

5

<div class="gallery-group-main">

{% galleryGroup '壁紙' '收藏的一些壁紙' '/Gallery/wallpaper' https://i.loli.net/2019/11/10/T7Mu8Aod3egmC4Q.png %}

{% galleryGroup '漫威' '關於漫威的圖片' '/Gallery/marvel' https://i.loli.net/2019/12/25/8t97aVlp4hgyBGu.jpg %}

{% galleryGroup 'OH MY GIRL' '關於OH MY GIRL的圖片' '/Gallery/ohmygirl' https://i.loli.net/2019/12/25/hOqbQ3BIwa6KWpo.jpg %}

</div>

壁紙

收藏的一些壁紙

漫威

關於漫威的圖片

OH MY GIRL

關於 OH MY GIRL 的圖片

Gallery 相冊

2.0.0 以上提供

區別於舊版的 Gallery 相冊，新的 Gallery 相冊會自動根據圖片長度進行排版，書寫也更加方便，與 markdown 格式一樣。可根據需要插入到相應的 md。

寫法:

Markdown

1

2

3

{% gallery %}

markdown 圖片格式

{% endgallery %}

例如

Markdown

1

2

3

4

5

6

7

8

9

10

{% gallery %}

![](https://i.loli.net/2019/12/25/Fze9jchtnyJXMHN.jpg)

![](https://i.loli.net/2019/12/25/ryLVePaqkYm4TEK.jpg)

![](https://i.loli.net/2019/12/25/gEy5Zc1Ai6VuO4N.jpg)

![](https://i.loli.net/2019/12/25/d6QHbytlSYO4FBG.jpg)

![](https://i.loli.net/2019/12/25/6nepIJ1xTgufatZ.jpg)

![](https://i.loli.net/2019/12/25/E7Jvr4eIPwUNmzq.jpg)

![](https://i.loli.net/2019/12/25/mh19anwBSWIkGlH.jpg)

![](https://i.loli.net/2019/12/25/2tu9JC8ewpBFagv.jpg)

{% endgallery %}

tag-hide

2.2.0 以上提供

請注意，tag-hide 內的標簽外掛 content 內都不建議有 h1 - h6 等標題。因為 Toc 會把隱藏內容標題也顯示出來，而且當滾動屏幕時，如果隱藏內容沒有顯示出來，會導致 Toc 的滾動出現異常。

如果你想把一些文字、內容隱藏起來，並提供按鈕讓用户點擊顯示。可以使用這個標籤外掛。

Inline

Block

Toggle

inline 在文本里面添加按鈕隱藏內容，只限文字

( content 不能包含英文逗號，可用 &sbquo;)

Markdown

1

{% hideInline content,display,bg,color %}

content: 文本內容

display: 按鈕顯示的文字 (可選)

bg: 按鈕的背景顏色 (可選)

color: 按鈕文字的顏色 (可選)

Demo

Markdown

1

2

3

哪個英文字母最酷？ {% hideInline 因為西裝褲(C裝酷),查看答案,#FF7242,#fff %}

門裏站着一個人? {% hideInline 閃 %}

哪個英文字母最酷？ 查看答案

門裏站着一個人？Click

mermaid

mermaid 標籤不允許嵌套於一些隱藏屬性的標籤外掛，例如: tag-hide 內的標籤外掛和 tabs 標籤外掛，這會導致 mermaid 圖示無法正常顯示，使用時請留意。

請不要壓縮 html 代碼，不然會導致 mermaid 顯示異常

使用 mermaid 標籤可以繪製 Flowchart（流程圖）、Sequence diagram（時序圖 ）、Class Diagram（類別圖）、State Diagram（狀態圖）、Gantt（甘特圖）和 Pie Chart（圓形圖），具體可以查看 mermaid 文檔

配置 butterfly.yml

Yaml

1

2

3

4

mermaid:

enable: true

# built-in themes: default/forest/dark/neutral

theme: default

寫法：

Markdown

1

2

3

{% mermaid %}

內容

{% endmermaid %}

例如：

Markdown

1

2

3

4

5

6

7

8

{% mermaid %}

pie

title Key elements in Product X

"Calcium" : 42.96

"Potassium" : 50.05

"Magnesium" : 10.01

"Iron" : 5

{% endmermaid %}

Tabs

移植於 next 主題

使用方法

Markdown

{% tabs Unique name, [index] %}

<!-- tab [Tab caption] [@icon] -->

Any content (support inline tags too).

<!-- endtab -->

{% endtabs %}

Unique name : Unique name of tabs block tag without comma.

Will be used in #id's as prefix for each tab with their index numbers.

If there are whitespaces in name, for generate #id all whitespaces will replaced by dashes.

Only for current url of post/page must be unique!

[index] : Index number of active tab.

If not specified, first tab (1) will be selected.

If index is -1, no tab will be selected. It's will be something like spoiler.

Optional parameter.

[Tab caption] : Caption of current tab.

If not caption specified, unique name with tab index suffix will be used as caption of tab.

If not caption specified, but specified icon, caption will empty.

Optional parameter.

[@icon] : FontAwesome icon name (full-name, look like 'fas fa-font')

Can be specified with or without space; e.g. 'Tab caption @icon' similar to 'Tab caption@icon'.

Optional parameter.

Demo 1 - 預設選擇第一個【默認】

Markdown

1

2

3

4

5

6

7

8

9

10

11

12

13

{% tabs test1 %}

<!-- tab -->

\*\*This is Tab 1.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 2.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 3.\*\*

<!-- endtab -->

{% endtabs %}

test1 1

test1 2

test1 3

This is Tab 1.

Demo 2 - 預設選擇 tabs

Markdown

1

2

3

4

5

6

7

8

9

10

11

12

13

{% tabs test2, 3 %}

<!-- tab -->

\*\*This is Tab 1.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 2.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 3.\*\*

<!-- endtab -->

{% endtabs %}

test2 1

test2 2

test2 3

This is Tab 3.

Demo 3 - 沒有預設值

Markdown

1

2

3

4

5

6

7

8

9

10

11

12

13

{% tabs test3, -1 %}

<!-- tab -->

\*\*This is Tab 1.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 2.\*\*

<!-- endtab -->

<!-- tab -->

\*\*This is Tab 3.\*\*

<!-- endtab -->

{% endtabs %}

test3 1

test3 2

test3 3

Demo 4 - 自定義 Tab 名 + 只有 icon + icon 和 Tab 名

Markdown

1

2

3

4

5

6

7

8

9

10

11

12

13

{% tabs test4 %}

<!-- tab 第一個Tab -->

\*\*tab名字為第一個Tab\*\*

<!-- endtab -->

<!-- tab @fab fa-apple-pay -->

\*\*只有圖標 沒有Tab名字\*\*

<!-- endtab -->

<!-- tab 炸彈@fas fa-bomb -->

\*\*名字+icon\*\*

<!-- endtab -->

{% endtabs %}

第一個 Tab

炸彈

tab 名字為第一個 Tab

Button

3.0 以上適用

使用方法：

Markdown

{% btn [url],[text],[icon],[color] [style] [layout] [position] [size] %}

[url] : 鏈接

[text] : 按鈕文字

[icon] : [可選] 圖標

[color] : [可選] 按鈕背景顔色(默認style時）

按鈕字體和邊框顔色(outline時)

default/blue/pink/red/purple/orange/green

[style] : [可選] 按鈕樣式 默認實心

outline/留空

[layout] : [可選] 按鈕佈局 默認為line

block/留空

[position] : [可選] 按鈕位置 前提是設置了layout為block 默認為左邊

center/right/留空

[size] : [可選] 按鈕大小

larger/留空

Demo

Markdown

1

2

3

4

5

This is my website, click the button {% btn 'http://www.jerryc.me',JerryC %}

This is my website, click the button {% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right %}

This is my website, click the button {% btn 'http://www.jerryc.me',JerryC,,outline %}

This is my website, click the button {% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline %}

This is my website, click the button {% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,larger %}

This is my website, click the button JerryC

This is my website, click the button JerryC

This is my website, click the button JerryC

This is my website, click the button JerryC

This is my website, click the button JerryC

Markdown

1

2

3

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,block %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,block center larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,block right outline larger %}

JerryC

JerryC

JerryC

more than one button in center

Markdown

1

2

3

4

5

6

7

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,blue larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,pink larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,red larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,purple larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,orange larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,green larger %}

JerryC JerryC JerryC JerryC JerryC JerryC JerryC

Markdown

1

2

3

4

5

6

7

8

9

<div class="btn-center">

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline blue larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline pink larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline red larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline purple larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline orange larger %}

{% btn 'http://www.jerryc.me',JerryC,far fa-hand-point-right,outline green larger %}

</div>

JerryC JerryC JerryC JerryC JerryC JerryC JerryC

配置文件説明

站點配置文件\_config.yml 是 hexo 工作目錄下的配置文件

butterfly.yml（如有創建）是 Butterfly 的配置文件，等同於主題目錄下的\_config.yml

評論

只能選擇一個評論服務商

Disqus

Disqusjs

livere（來必力）

Gitalk

Valine

Utterances

Facebook Comments

註冊 disqus，配置你自己的 disqus，然後在 Butterfly 裏開啟它。

配置 butterfly.yml

Yaml

1

2

3

4

disqus:

enable: true

shortname:

count: false # top\_img顯示評論數

在綫聊天

從 3.0 開始，Butterfly 主題內置了 4 種在綫聊天工具。你可以選擇開啟一種，方便你與訪客的交流。

通用設置

chatra

tidio

daovoice

Gitter

這些工具都提供了一個按鈕可以打開 / 關閉聊天窗口。

主題也提供了一個集合主題特色的按鈕來替換這些工具本身的按鈕，這個聊天按鈕將會出現在右下角裏。

你只需要把 chat\_btn 打開就行。

配置 butterfly.yml

Yaml

1

2

3

# Chat Button [recommend]

# It will create a button in the bottom right corner of website, and hide the origin button

chat\_btn: true

為了不影響訪客的體驗，主題提供一個 chat\_hide\_show 配置

設為 true 後，使用工具提供的按鈕時，只有向上滾動才會顯示聊天按鈕，向下滾動時會隱藏按鈕。

配置 butterfly.yml

Yaml

1

2

# The origin chat button is displayed when scrolling up, and the button is hidden when scrolling down

chat\_hide\_show: true

如果使用工具自帶的聊天按鈕，按鈕位置可能會遮擋右下角圖標，請配置 rightside-bottom 調正右下角圖標位置

分享

只能選擇一個分享服務商

AddThis

Sharejs

Addtoany

訪問 AddThis 官網

找到你的 pub-id

配置 butterfly.yml

Yaml

1

2

3

addThis:

enable: true # or false

pubid: 你的pub-id

搜索系統

Algolia

本地搜索

你需要安裝 hexo-algolia 或 hexo-algoliasearch. 根據它們的説明文檔去做相應的配置。

配置 butterfly.yml

Yaml

1

2

3

4

5

6

7

8

9

algolia\_search:

enable: true

hits:

per\_page: 6

labels:

input\_placeholder: Search for Posts

hits\_empty: "We didn't find any results for the search: ${query}" # if there are no result

hits\_stats: "${hits} results found in ${time} ms"

### 網站驗證

如果需要搜索引擎收錄網站，可能需要登錄對應搜索引擎的管理平台進行提交。

各自的驗證碼可從各自管理平台拿到

配置 butterfly.yml

Yaml

# Google Webmaster tools verification setting

# See: https://www.google.com/webmasters/

google\_site\_verification:

# Bing Webmaster tools verification setting

# See: https://www.bing.com/webmaster/

bing\_site\_verification:

# Baidu Webmaster tools verification setting

# See: https://ziyuan.baidu.com/site/

baidu\_site\_verification:

# 360 Webmaster tools verification setting

# see http://zhanzhang.so.com/

qihu\_site\_verification:

分析統計

百度統計

谷歌分析

騰訊分析

登錄百度統計的官方網站

找到你百度統計的統計代碼

配置 butterfly.yml

Yaml

1

baidu\_analytics: 你的代碼

廣告

谷歌廣告

手動廣告配置

主題已集成谷歌廣告（自動廣告）

配置 butterfly.yml

Yaml

1

2

3

4

5

google\_adsense:

enable: true

js: https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js

client: # 填入個人識別碼

enable\_page\_level\_ads: true

Math 數學

MathJax

KaTeX

建議使用 KaTex 獲得更好的效果，下文有介紹！

配置 butterfly.yml:

Yaml

1

2

3

4

5

mathjax:

enable: true

# true 表示每一頁都加載mathjax.js

# false 需要時加載，須在使用的Markdown Front-matter 加上 mathjax: true

per\_page: false

如果 per\_page 設為 true, 則每一頁都會加載 Mathjax 服務。設為 false，則需要在文章 Front-matter 添加 mathjax: true, 對應的文章才會加載 Mathjax 服務。

然後你需要修改一下默認的 markdown 渲染引擎來實現 MathJax 的效果。

查看: hexo-renderer-kramed

以下操作在你 hexo 博客的目錄下 (不是 Butterfly 的目錄！):

效果：

美化 / 特效

自定義主題色

可以修改大部分 UI 顏色

配置 butterfly.yml，比如：

顏色值必須被雙引號包裹，就像 "#000" 而不是#000。否則將會在構建的時候報錯！

Yaml

theme\_color:

enable: true

main: "#49B1F5"

paginator: "#00c4b6"

button\_hover: "#FF7242"

text\_selection: "#00c4b6"

link\_color: "#99a9bf"

meta\_color: "#858585"

hr\_color: "#A4D8FA"

code\_foreground: "#F47466"

code\_background: "rgba(27, 31, 35, .05)"

toc\_color: "#00c4b6"

blockquote\_padding\_color: "#49b1f5"

blockquote\_background\_color: "#49b1f5"

網站背景

默認顯示白色，可設置圖片或者顏色

配置 butterfly.yml

Yaml

1

2

3

4

# 圖片格式 background: url(http://xxxxxx.com/xxx.jpg)

# 顏色 background: '#49B202'

# 留空 顯示白色

background:

留意: 如果你的網站根目錄不是’/‘, 使用本地圖片時，需加上你的根目錄。

例如：網站是 https://yoursite.com/blog, 引用一張 img/xx.png 圖片，則設置 background 為 `url (/blog/img/xx.png)

background:’#49B202’

background: url(https://i.loli.net/2019/09/09/5oDRkWVKctx2b6A.png)

footer 背景

footer 的背景會與 top\_img 的一致，當設置 false 時，將與主題色一致。

配置 butterfly.yml

Yaml

1

2

# footer是否顯示圖片背景(與top\_img一致)

footer\_bg: true

true

打字效果

打字效果 activate-power-mode

配置 butterfly.yml

Yaml

1

2

3

4

5

6

# Typewriter Effect (打字效果)

# https://github.com/disjukr/activate-power-mode

activate\_power\_mode:

enable: true

colorful: true # open particle animation (冒光特效)

shake: true # open shake (抖動特效)

背景特效

靜止綵帶

動態綵帶

canvas-nest

好看的綵帶背景，可設置每次刷新更換綵帶，或者每次點擊更換綵帶

配置 butterfly.yml

Yaml

1

2

3

4

5

6

7

canvas\_ribbon:

enable: false

size: 150

alpha: 0.6

zIndex: -1

click\_to\_change: false #設置是否每次點擊都更換綵帶

mobile: false # false 手機端不顯示 true 手機端顯示

相關配置可查看 canvas\_ribbon

鼠標點擊效果

煙花

愛心

文字

zIndex 建議只在 -1 和 9999 上選

-1 代表煙火效果在底部

9999 代表煙火效果在前面

配置 butterfly.yml

Yaml

1

2

3

fireworks:

enable: true

zIndex: 9999 # -1 or 9999

頁面美化

會改變 ol、ul、h1-h5 的樣式

field 配置生效的區域

post 只在文章頁生效

site 在全站生效

配置 butterfly.yml

Yaml

1

2

3

4

5

6

# 美化頁面顯示

beautify:

enable: true

field: site # site/post

title-prefix-icon: '\f0c1'

title-prefix-icon-color: "#F47466"

title-prefix-icon 填寫的是 fontawesome 的 icon 的 Unicode 數。

未開啟美化

開啟美化

自定義字體

全局字體

可自行設置字體的 font-family

如不需要配置，請留空

配置 butterfly.yml

Yaml

1

2

3

4

5

# Global font settings

# Don't modify the following settings unless you know how they work (非必要不要修改)

font:

font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Helvetica Neue", Lato, Roboto, "PingFang SC", "Microsoft JhengHei", "Microsoft YaHei", sans-serif

code-font: consolas, Menlo, "PingFang SC", "Microsoft JhengHei", "Microsoft YaHei", sans-serif

Blog 標題字體

可自行設置字體的 font-family

如不需要配置，請留空。

如不需要使用網絡字體，只需要把 font\_link 留空就行

配置 butterfly.yml

Yaml

# Font settings for the site title and site subtitle

# 左上角網站名字 主頁居中網站名字

blog\_title\_font:

font\_link: https://fonts.googleapis.com/css?family=Titillium+Web&display=swap

font-family: Titillium Web, 'PingFang SC', 'Hiragino Sans GB', 'Microsoft JhengHei', 'Microsoft YaHei', sans-serif

網站副標題

可設置主頁中顯示的網站副標題或者喜歡的座右銘。

配置 butterfly.yml

Yaml

# 主頁subtitle

subtitle:

enable: true

# 打字效果

effect: true

# 循環或者只打字一次

loop: false

# source調用第三方服務

# source: false 關閉調用

# source: 1 調用搏天api的隨機語錄（簡體） https://api.btstu.cn/

# source: 2 調用一言網的一句話（簡體） https://hitokoto.cn/

# source: 3 調用一句網（簡體） http://yijuzhan.com/

# source: 4 調用今日詩詞（簡體） https://www.jinrishici.com/

# subtitle 會先顯示 source , 再顯示 sub 的內容

source: false

# 如果有英文逗號' , ',請使用轉義字元 &#44;

# 如果有英文雙引號' " ',請使用轉義字元 &quot;

# 開頭不允許轉義字元，如需要，請把整個句子用雙引號包住

# 例如 ”&quotNever put off till tomorrow what you can do today&quot"

# 如果關閉打字效果，subtitle只會顯示sub的第一行文字

sub:

- 今日事&#44;今日畢

- Never put off till tomorrow what you can do today

主頁 top\_img 顯示大小

適用於 版本號 >= V1.2.0

默認的顯示為全屏。site-info 的區域會居中顯示

Yaml

1

2

3

4

5

# 主頁設置

# 默認top\_img全屏，site\_info在中間

# 使用默認, 都無需填寫（建議默認）

index\_site\_info\_top: # 主頁標題距離頂部距離 例如 300px/300em/300rem/10%

index\_top\_img\_height: #主頁top\_img高度 例如 300px/300em/300rem 不能使用百分比

注意：index\_top\_img\_height 的值不能使用百分比。

2 個都不填的話，會使用默認值

舉例，當

Yaml

1

index\_top\_img\_height: 400px

效果

頁面加載動畫 preloader

當進入網頁時，因為加載速度的問題，可能會導致 top\_img 圖片出現斷層顯示，或者網頁加載不全而出現等待時間，開啟 preloader 後，會顯示加載動畫，等頁面加載完，加載動畫會消失。

配置 butterly.yml

Yaml

1

2

# 加載動畫 Loading Animation

preloader: true

PWA

要為 Butterfly 配上 PWA 特性，你需要如下幾個步驟:

打開 hexo 工作目錄

npm install hexo-offline --save 或者 yarn add hexo-offline

修改\_config.yml 在站點\_config.yml 中增加以下內容。

Yaml

# offline config passed to sw-precache.

offline:

maximumFileSizeToCacheInBytes: 10485760 # 緩存的最大文件大小，以字節為單位

staticFileGlobs:

- public/\*\*/\*.{js,html,css,png,jpg,gif,svg,webp,eot,ttf,woff,woff2}

# 靜態文件合集，如果你的站點使用了例如webp格式的文件，請將文件類型添加進去。

stripPrefix: public

verbose: true

runtimeCaching:

# CDNs - should be cacheFirst, since they should be used specific versions so should not change

- urlPattern: /\* # 如果你需要加載CDN資源，請配置該選項，如果沒有，可以不配置。

handler: cacheFirst

options:

origin: your\_websie\_url # 可替換成你的 url

更多內容請查看 hexo-offline 的官方文檔

在 butterfly.yml 中開啟 pwa 選項。

Yaml

1

2

3

4

5

6

7

8

pwa:

enable: true

manifest: /img/pwa/manifest.json

theme\_color: "#fff"

apple\_touch\_icon: /img/pwa/apple-touch-icon.png

favicon\_32\_32: /img/pwa/32.png

favicon\_16\_16: /img/pwa/16.png

mask\_icon: /img/pwa/safari-pinned-tab.svg

在創建 source/ 目錄中創建 manifest.json 文件。

Json

{

"name": "string", //應用全稱

"short\_name": "Junzhou", //應用簡稱

"theme\_color": "#49b1f5", //匹配瀏覽器的地址欄顏色

"background\_color": "#49b1f5",//加載應用時的背景色

"display": "standalone",//首選顯示模式 其他選項有：fullscreen,minimal-ui,browser

"scope": "/",

"start\_url": "/",

"icons": [ //該數組指定icons圖標參數，用來時適配不同設備（需為png，至少包含一個192px\*192px的圖標）

{

"src": "images/pwaicons/36.png", //圖標文件的目錄，需在source/目錄下自行創建。

"sizes": "36x36",

"type": "image/png"

},

{

"src": "images/pwaicons/48.png",

"sizes": "48x48",

"type": "image/png"

},

{

"src": "images/pwaicons/72.png",

"sizes": "72x72",

"type": "image/png"

},

{

"src": "images/pwaicons/96.png",

"sizes": "96x96",

"type": "image/png"

},

{

"src": "images/pwaicons/144.png",

"sizes": "144x144",

"type": "image/png"

},

{

"src": "images/pwaicons/192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "images/pwaicons/512.png",

"sizes": "512x512",

"type": "image/png"

}

],

"splash\_pages": null //配置自定義啟動動畫。

}

你也可以通過 Web App Manifest 快速創建 manifest.json。（Web App Manifest 要求至少包含一個 512\*512 像素的圖標）

可以通過 Chrome 插件 Lighthouse 檢查 PWA 配置是否生效以及配置是否正確。

打開博客頁面

啟動 Lighthouse 插件 (Lighthouse 插件要求至少包含一個 512\*512 像素的圖標)

關於 PWA（漸進式增強 Web 應用）的更多內容請參閲 Google Tools for Web Developers

字數統計

要為 Butterfly 配上字數統計特性，你需要如下幾個步驟:

打開 hexo 工作目錄

npm install hexo-wordcount --save or yarn add hexo-wordcount

配置 butterfly.yml:

Yaml

wordcount:

enable: true

post\_wordcount: true

min2read: true

total\_wordcount: true

圖片大圖查看模式

fancybox

medium\_zoom

配置 butterfly.yml

Yaml

1

2

# fancybox http://fancyapps.com/fancybox/3/

fancybox: true

Snackbar 彈窗

Snackbar 彈窗，根據自己愛好開啟

配置 butterfly.yml

Yaml

1

2

3

4

5

6

7

8

9

# Snackbar 彈窗

# https://github.com/polonel/SnackBar

# position 彈窗位置

# 可選 top-left / top-center / top-right / bottom-left / bottom-center / bottom-right

snackbar:

enable: true

position: bottom-left

bg\_light: '#49b1f5' #light mode時彈窗背景

bg\_dark: '#2d3035' #dark mode時彈窗背景

未開啟 Snackbar

開啟 Snackbar

豆瓣

很多人使用 hexo-douban 這款插件。但是該插件是直接生成相應的網頁，而不會去讀取對應的 markdown 文件。因此，可在設置裏進行相應的配置。

配置 butterfly.yml

Markdown

1

2

3

4

5

6

# 如果你有使用hexo-douban，可配置這個

douban:

meta: true

movies\_img: https://cdn.jsdelivr.net/gh/jerryc127/butterfly\_cdn@2.1.0/top\_img/movie.jpg

books\_img:

games\_img:

由於豆瓣開啟了反盜鏈，因此，一些瀏覽器（safari）會無法顯示圖片，可把 meta 設為 true。但此操作會導致一些插件失效（例如：不蒜子）。

Inject

2.3.0 以上支持

如想添加額外的 js/css/meta 等等東西，可以在 Inject 裏添加，支持添加到 head (</body> 標籤之前) 和 bottom (</html> 標籤之前)。

請注意：以標準的 html 格式添加內容

例如

Yaml

1

2

3

4

5

inject:

head:

- <link rel="stylesheet" href="/self.css">

bottom:

- <script src="xxxx"></script>

留意: 如果你的網站根目錄不是’/‘, 使用本地圖片時，需加上你的根目錄。

例如：網站是 https://yoursite.com/blog, 引用 css/xx.css，則設置為 <link rel="stylesheet" href="/blog/css/xx.css">

CDN

配置文件中最後一部分 CDN，裏面是主題所引用到的文件，可自行配置 CDN。（非必要請勿修改，配置後請確認鏈接是否能訪問）

以下是一些 CDN 提供商：

jsdelivr 👍主題使用

bootcdn

cdnjs

……

運行後網頁顯示代碼

頁面只顯示 extends includes/layout.pug block content #recent-posts.recent-posts include includes/recent-posts.pug include includes/pagination.pug #aside\_content.aside\_content include includes/aside.pug

請下載安裝：npm install hexo-renderer-pug hexo-renderer-stylus --save or `yarn add hexo-renderer-pug hexo-renderer-stylus

配置友情鏈接報錯

配置友情鏈接頁面時出現報錯

Code

ERROR D:\Desktop\orxing-blog\themes\Butterfly\layout\flink.pug:2

1| .flink

> 2| each i in site.data.link

3| p.comment-word= i.class\_name

4| .post-cards

5| ul.md-links

Cannot read property 'length' of undefined

TypeError: D:\Desktop\orxing-blog\themes\Butterfly\layout\flink.pug:2

1| .flink

> 2| each i in site.data.link

3| p.comment-word= i.class\_name

4| .post-cards

5| ul.md-links

Cannot read property 'length' of undefined

at eval (eval at wrap (D:\Desktop\orxing-blog\node\_modules\pug-runtime\wrap.js:6:10), <anonymous>:1890:32)

at eval (eval at wrap (D:\Desktop\orxing-blog\node\_modules\pug-runtime\wrap.js:6:10), <anonymous>:2017:4)

at template (eval at wrap (D:\Desktop\orxing-blog\node\_modules\pug-runtime\wrap.js:6:10), <anonymous>:10152:72)

at Theme.\_View.View.\_compiled (D:\Desktop\orxing-blog\node\_modules\hexo\lib\theme\view.js:123:48)

at Theme.\_View.View.View.render (D:\Desktop\orxing-blog\node\_modules\hexo\lib\theme\view.js:29:15)

at D:\Desktop\orxing-blog\node\_modules\hexo\lib\hexo\index.js:349:21

at tryCatcher (D:\Desktop\orxing-blog\node\_modules\bluebird\js\release\util.js:16:23)

at D:\Desktop\orxing-blog\node\_modules\bluebird\js\release\method.js:15:34

at RouteStream.\_read (D:\Desktop\orxing-blog\node\_modules\hexo\lib\hexo\router.js:123:3)

at RouteStream.Readable.read (\_stream\_readable.js:457:10)

at resume\_ (\_stream\_readable.js:936:12)

at processTicksAndRejections (internal/process/task\_queues.js:84:9)

請檢查 link.yml 文檔內代碼的空格

升級最新版後運行報錯

升級最新版本 hexo g 後報錯

Code

INFO Deleted database.

INFO Start processing

FATAL Something's wrong. Maybe you can find the solution here: https://hexo.io/docs/troubleshooting.html

TypeError: Cannot read property 'enable' of undefined

at Hexo.<anonymous> (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/themes/Butterfly/scripts/post-lazyload.js:5:23)

at Hexo.tryCatcher (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/util.js:16:23)

at Hexo.<anonymous> (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/method.js:15:34)

at /Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/hexo/lib/extend/filter.js:60:50

at tryCatcher (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/util.js:16:23)

at Object.gotValue (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/reduce.js:155:18)

at Object.gotAccum (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/reduce.js:144:25)

at Object.tryCatcher (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/util.js:16:23)

at Promise.\_settlePromiseFromHandler (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/promise.js:517:31)

at Promise.\_settlePromise (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/promise.js:574:18)

at Promise.\_settlePromiseCtx (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/promise.js:611:10)

at \_drainQueueStep (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/async.js:142:12)

at \_drainQueue (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/async.js:131:9)

at Async.\_drainQueues (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/async.js:147:5)

at Immediate.Async.drainQueues [as \_onImmediate] (/Users/qinkangdeid/Nextcloud/work/codes/personal/github/personal/qinkangdeid.github.io/node\_modules/bluebird/js/release/async.js:17:14)

at processImmediate (internal/timers.js:439:21)

請參照最新版的\_config.yml, 比對後，把缺的配置複製到 butterfly.yml 去

wordcount 沒找到

報錯 wordcount is not a function

請檢查是否安裝了 wordcount 插件 npm i --save hexo-wordcount

升級 2.0.0 後運行報錯

舊版本升級到 2.0.0 後報錯

Code

INFO Start processing

FATAL Something’s wrong. Maybe you can find the solution here: https://hexo.io/docs/troubleshooting.html

Template render error: (unknown path)

unexpected end of file

at Object.prettifyError (F:\hexo\nodemodules\nunjucks\src\lib.js:36:11)

at Template.render (F:\hexo\node\_modules\nunjucks\src\environment.js:542:21)

at Environment.renderString (F:\hexo\node\_modules\nunjucks\src\environment.js:380:17)

at Promise.fromCallback.cb (F:\hexo\node\_modules\hexo\lib\extend\tag.js:123:48)

at tryCatcher (F:\hexo\node\_modules\bluebird\js\release\util.js:16:23)

at Function.Promise.fromNode.Promise.fromCallback (F:\hexo\node\_modules\bluebird\js\release\promise.js:185:30)

at Tag.render (F:\hexo\node\_modules\hexo\lib\extend\tag.js:123:18)

at Object.onRenderEnd (F:\hexo\node\_modules\hexo\lib\hexo\post.js:280:20)

at Promise.then.then.result (F:\hexo\node\_modules\hexo\lib\hexo\render.js:64:19)

at tryCatcher (F:\hexo\node\_modules\bluebird\js\release\util.js:16:23)

at Promise.settlePromiseFromHandler (F:\hexo\nodemodules\bluebird\js\release\promise.js:517:31)

2.0.0 版本以上刪掉了 gallery，而引入新的 gallery。所以如果有使用舊版，需刪掉或者更改寫法。

代碼渲染與實際不同

2.1.0 以下版本會出現的問題

代碼渲染與實際不同

Code

1

<div>aaa</div>

渲染結果:

cheerio 版本錯誤，請安裝 0.22.0 版本

npm install cheerio@0.22.0 –save

搜索欄在底部

hexo deploy 後搜索欄在頁面底部

生成前先 hexo clean

本地可以正常運行，但是 push 上去後出錯 / 缺失 / 無效

清理瀏覽器緩存

如果 1 無效，請確認上傳時是否有運行 hexo clean

已在 Hexo 的配置文件設置了語言，爲什麽導航欄仍然是英文

請在導航的配置裏，自己修改成想要的中文

Yaml

1

2

3

4

首頁: / || fa fa-home

時間軸: /archives/ || fa fa-archive

標籤: /tags/ || fa fa-tags

分類: /categories/ || fa fa-folder-open

運行報錯 Cannot read property ‘bind’ of undefined

Powershell

1

TypeError: Cannot read property 'bind' of undefined

把 Hexo 升級到 4.0 以上版本

建議

不要把個人需要的文件 / 圖片放在主題 source 文件夾裏，因為在升級主題的過程中，可能會把文件覆蓋刪除了。

在 Hexo 根目錄的 source 文件夾裏，創建一個文件夾來放置個人文件 / 圖片。文件夾不能命名為 css、js 和 img

引用文件直接為 /文件夾名稱/文件名

升級完主題，記得把新添加的配置複製到 butterfly.yml 去（如有）

Gulp 壓縮

Gulp 是一款自動化構建的工具，擁有眾多的插件。而我們只需要使用到幾個插件來壓縮 Html/css/js。

安裝 Gulp

Powershell

1

npm install --global gulp-cli

插件安裝

壓縮 HTML

可以使用 gulp-htmlclean 和 gulp-htmlmin 來壓縮 HTML

Powershell

npm install gulp-htmlclean --save-dev

npm install --save gulp-htmlmin

壓縮 CSS

可以使用 gulp-clean-css 來壓縮 CSS

Powershell

1

npm install gulp-clean-css --save-dev

壓縮 JS

由於 Butterfly 主題中的 JS 使用到了部分 ES6 語法，因此不能只使用 gulp-uglify 來壓縮，還需要搭配其它的插件。兩種搭配都可以有效的壓縮 JS 代碼，選一種適合自己的就行。

terser 是 ES6+ 的 JavaScript 解析器

gulp-babel 是一個 JavaScript 轉換編譯器，可以把 ES6 轉換成 ES5

gulp-uglify + terser

Powershell

1

2

npm install --save-dev gulp-uglify

npm install terser

gulp-uglify + gulp-babel

Powershell

1

2

npm install --save-dev gulp-uglify

npm install --save-dev gulp-babel @babel/core @babel/preset-env

壓縮圖片

可以使用 gulp-imagemin 來壓縮圖片

Powershell

1

npm install --save-dev gulp-imagemin

創建 gulpfile 文件

在 Hexo 的根目錄，創建一個 gulpfile.js 文件

Js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

var gulp = require('gulp')

var cleanCSS = require('gulp-clean-css')

var htmlmin = require('gulp-htmlmin')

var htmlclean = require('gulp-htmlclean')

var imagemin = require('gulp-imagemin')

// tester (如果使用tester,把下面4行前面的//去掉)

// var uglifyjs = require('terser')

// var composer = require('gulp-uglify/composer')

// var pump = require('pump')

// var minify = composer(uglifyjs, console)

// babel (如果不是使用bebel,把下面兩行註釋掉)

var uglify = require('gulp-uglify')

var babel = require('gulp-babel')

// minify js - babel（ 如果不是使用bebel,把下面註釋掉）

gulp.task('compress', () =>

gulp.src(['./public/\*\*/\*.js', '!./public/\*\*/\*.min.js'])

.pipe(babel({

presets: ['@babel/preset-env']

}))

.pipe(uglify().on('error', function (e) {

console.log(e)

}))

.pipe(gulp.dest('./public'))

)

// minify js - tester (如果使用tester,把下面前面的//去掉)

// gulp.task('compress', function (cb) {

// var options = {}

// pump([

// gulp.src(['./public/\*\*/\*.js', '!./public/\*\*/\*.min.js']),

// minify(options),

// gulp.dest('./public')

// ],

// cb

// )

// })

// css

gulp.task('minify-css', () => {

return gulp.src('./public/\*\*/\*.css')

.pipe(cleanCSS())

.pipe(gulp.dest('./public'))

})

// 壓縮 public 目錄內 html

gulp.task('minify-html', () => {

return gulp.src('./public/\*\*/\*.html')

.pipe(htmlclean())

.pipe(htmlmin({

removeComments: true, // 清除 HTML 註釋

collapseWhitespace: true, // 壓縮 HTML

collapseBooleanAttributes: true, // 省略布爾屬性的值 <input checked="true"/> ==> <input />

removeEmptyAttributes: true, // 刪除所有空格作屬性值 <input id="" /> ==> <input />

removeScriptTypeAttributes: true, // 刪除 <script> 的 type="text/javascript"

removeStyleLinkTypeAttributes: true, // 刪除 <style> 和 <link> 的 type="text/css"

minifyJS: true, // 壓縮頁面 JS

minifyCSS: true, // 壓縮頁面 CSS

minifyURLs: true

}))

.pipe(gulp.dest('./public'))

})

// 壓縮 public/uploads 目錄內圖片

gulp.task('minify-images', async () => {

gulp.src('./public/img/\*\*/\*.\*')

.pipe(imagemin({

optimizationLevel: 5, // 類型：Number 預設：3 取值範圍：0-7（優化等級）

progressive: true, // 類型：Boolean 預設：false 無失真壓縮jpg圖片

interlaced: false, // 類型：Boolean 預設：false 隔行掃描gif進行渲染

multipass: false // 類型：Boolean 預設：false 多次優化svg直到完全優化

}))

.pipe(gulp.dest('./public/img'))

})

// 執行 gulp 命令時執行的任務

gulp.task('default', gulp.parallel(

'compress', 'minify-css', 'minify-html', 'minify-images'

))

注意： 如果有使用到 Butterfly 主題提供的 mermaid 標籤外掛，那需要把 52 行.pipe(htmlclean()) 註釋掉，同時，把 55 行的 collapseWhitespace: true, 改為 false

運行

在 hexo g 之後運行 gulp 就行。

Powershell

1

gulp

PWA

這是另一種實現 PWA 的方法，使用這個方法之前，請先卸載掉其它的 PWA 插件。

這部分文章參考利用 Workbox 實現博客的 PWA

據維基百科介紹

漸進式網絡應用程式（英語：Progressive Web Apps，簡稱：PWA）是一種普通網頁或網站架構起來的網絡應用程式，但它可以以傳統應用程式或原生移動應用程式形式展示給用户。這種應用程式形態視圖將目前最為現代化的瀏覽器提供的功能與行動裝置的體驗優勢相結合。

當你的網站實現了 PWA，那就代表了

用户可以添加你的博客到電腦╱手機的桌面，以原生應用般的方式瀏覽你的博客

用户可以更快速地瀏覽你的博客

用户可以離線瀏覽你的博客

Hexo 已經有很多插件可以實現 PWA，下面是另一種實現方法，需要有 Gulp 就行。這種方法也可實現彈窗提醒用户刷新網站（當網站有更新時）

此方法是使用 Service Worker。我們使用 Workbox 這個工具生成 sw.js 以快速實現 Service Worker ，並實現頁面的預緩存和頁面更新後的提醒功能。

開啟設置和配置 manifest.json

在 butterfly.yml 中開啟 pwa 選項

Yaml

1

2

3

4

5

6

7

8

pwa:

enable: true

manifest: /img/pwa/manifest.json

theme\_color: "#fff"

apple\_touch\_icon: /img/pwa/apple-touch-icon.png

favicon\_32\_32: /img/pwa/32.png

favicon\_16\_16: /img/pwa/16.png

mask\_icon: /img/pwa/safari-pinned-tab.svg

在 Hexo 的 source 目錄中創建 manifest.json 文件。

Json

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

{

"name": "string", //應用全稱

"short\_name": "Junzhou", //應用簡稱

"theme\_color": "#49b1f5", //匹配瀏覽器的地址欄顏色

"background\_color": "#49b1f5",//加載應用時的背景色

"display": "standalone",//首選顯示模式 其他選項有：fullscreen,minimal-ui,browser

"scope": "/",

"start\_url": "/",

"icons": [ //該數組指定icons圖標參數，用來時適配不同設備（需為png，至少包含一個192px\*192px的圖標）

{

"src": "images/pwaicons/36.png", //圖標文件的目錄，需在source/目錄下自行創建。

"sizes": "36x36",

"type": "image/png"

},

{

"src": "images/pwaicons/48.png",

"sizes": "48x48",

"type": "image/png"

},

{

"src": "images/pwaicons/72.png",

"sizes": "72x72",

"type": "image/png"

},

{

"src": "images/pwaicons/96.png",

"sizes": "96x96",

"type": "image/png"

},

{

"src": "images/pwaicons/144.png",

"sizes": "144x144",

"type": "image/png"

},

{

"src": "images/pwaicons/192.png",

"sizes": "192x192",

"type": "image/png"

},

{

"src": "images/pwaicons/512.png",

"sizes": "512x512",

"type": "image/png"

}

],

"splash\_pages": null //配置自定義啟動動畫。

}

你也可以通過 Web App Manifest 快速創建 manifest.json。（Web App Manifest 要求至少包含一個 512\*512 像素的圖標）

安裝插件

在命令行中輸入安裝插件

Powershell

1

npm install workbox-build gulp --save-dev

創建 gulpfile.js 文件

在 Hexo 的根目錄，創建一個 gulpfile.js 文件

Js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

const gulp = require("gulp");

const workbox = require("workbox-build");

gulp.task('generate-service-worker', () => {

return workbox.injectManifest({

swSrc: './sw-template.js',

swDest: './public/sw.js',

globDirectory: './public',

globPatterns: [

"\*\*/\*.{html,css,js,json,woff2}"

],

modifyURLPrefix: {

"": "./"

}

});

});

gulp.task("build", gulp.series("generate-service-worker"));

創建 sw-template.js 文件

在 Hexo 的根目錄，創建一個 sw-template.js 文件

Js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

const workboxVersion = '5.1.3';

importScripts(`https://storage.googleapis.com/workbox-cdn/releases/${workboxVersion}/workbox-sw.js`);

workbox.core.setCacheNameDetails({

prefix: "your name"

});

workbox.core.skipWaiting();

workbox.core.clientsClaim();

workbox.precaching.precacheAndRoute(self.\_\_WB\_MANIFEST);

workbox.precaching.cleanupOutdatedCaches();

// Images

workbox.routing.registerRoute(

/\.(?:png|jpg|jpeg|gif|bmp|webp|svg|ico)$/,

new workbox.strategies.CacheFirst({

cacheName: "images",

plugins: [

new workbox.expiration.ExpirationPlugin({

maxEntries: 1000,

maxAgeSeconds: 60 \* 60 \* 24 \* 30

}),

new workbox.cacheableResponse.CacheableResponsePlugin({

statuses: [0, 200]

})

]

})

);

// Fonts

workbox.routing.registerRoute(

/\.(?:eot|ttf|woff|woff2)$/,

new workbox.strategies.CacheFirst({

cacheName: "fonts",

plugins: [

new workbox.expiration.ExpirationPlugin({

maxEntries: 1000,

maxAgeSeconds: 60 \* 60 \* 24 \* 30

}),

new workbox.cacheableResponse.CacheableResponsePlugin({

statuses: [0, 200]

})

]

})

);

// Google Fonts

workbox.routing.registerRoute(

/^https:\/\/fonts\.googleapis\.com/,

new workbox.strategies.StaleWhileRevalidate({

cacheName: "google-fonts-stylesheets"

})

);

workbox.routing.registerRoute(

/^https:\/\/fonts\.gstatic\.com/,

new workbox.strategies.CacheFirst({

cacheName: 'google-fonts-webfonts',

plugins: [

new workbox.expiration.ExpirationPlugin({

maxEntries: 1000,

maxAgeSeconds: 60 \* 60 \* 24 \* 30

}),

new workbox.cacheableResponse.CacheableResponsePlugin({

statuses: [0, 200]

})

]

})

);

// Static Libraries

workbox.routing.registerRoute(

/^https:\/\/cdn\.jsdelivr\.net/,

new workbox.strategies.CacheFirst({

cacheName: "static-libs",

plugins: [

new workbox.expiration.ExpirationPlugin({

maxEntries: 1000,

maxAgeSeconds: 60 \* 60 \* 24 \* 30

}),

new workbox.cacheableResponse.CacheableResponsePlugin({

statuses: [0, 200]

})

]

})

);

workbox.googleAnalytics.initialize();

把 prefix 修改為你博客的名字（英文），如果你想用其它緩存策略，請自行查看相關文檔

添加 js 進主題

配置 butterfly.yml, 添加需要的 css 和 js

Yaml

1

2

3

4

5

inject:

head:

- '<style type="text/css">.app-refresh{position:fixed;top:-2.2rem;left:0;right:0;z-index:99999;padding:0 1rem;font-size:15px;height:2.2rem;transition:all .3s ease}.app-refresh-wrap{display:flex;color:#fff;height:100%;align-items:center;justify-content:center}.app-refresh-wrap a{color:#fff;text-decoration:underline;cursor:pointer}</style>'

bottom:

- '<div class="app-refresh" id="app-refresh"> <div class="app-refresh-wrap"> <label>✨ 網站已更新最新版本 👉</label> <a href="javascript:void(0)" onclick="location.reload()">點擊刷新</a> </div></div><script>function showNotification(){if(GLOBAL\_CONFIG.Snackbar){var t="light"===document.documentElement.getAttribute("data-theme")?GLOBAL\_CONFIG.Snackbar.bgLight:GLOBAL\_CONFIG.Snackbar.bgDark,e=GLOBAL\_CONFIG.Snackbar.position;Snackbar.show({text:"已更新最新版本",backgroundColor:t,duration:5e5,pos:e,actionText:"點擊刷新",actionTextColor:"#fff",onActionClick:function(t){location.reload()}})}else{var o=`top: 0; background: ${"light"===document.documentElement.getAttribute("data-theme")?"#49b1f5":"#1f1f1f"};`;document.getElementById("app-refresh").style.cssText=o}}"serviceWorker"in navigator&&(navigator.serviceWorker.controller&&navigator.serviceWorker.addEventListener("controllerchange",function(){showNotification()}),window.addEventListener("load",function(){navigator.serviceWorker.register("/sw.js")}));</script>'

上面的代碼是壓縮過的，具體相關代碼如下，可供理解。

Html

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

<div class="app-refresh" id="app-refresh">

<div class="app-refresh-wrap">

<label>✨ 網站已更新最新版本 👉</label>

<a href="javascript:void(0)" onclick="location.reload()">點擊刷新</a>

</div>

</div>

<script>

if ('serviceWorker' in navigator) {

if (navigator.serviceWorker.controller) {

navigator.serviceWorker.addEventListener('controllerchange', function () {

showNotification()

})

}

window.addEventListener('load', function () {

navigator.serviceWorker.register('/sw.js')

})

}

function showNotification() {

if (GLOBAL\_CONFIG.Snackbar) {

var snackbarBg =

document.documentElement.getAttribute('data-theme') === 'light'

? GLOBAL\_CONFIG.Snackbar.bgLight

: GLOBAL\_CONFIG.Snackbar.bgDark

var snackbarPos = GLOBAL\_CONFIG.Snackbar.position

Snackbar.show({

text: '已更新最新版本',

backgroundColor: snackbarBg,

duration: 500000,

pos: snackbarPos,

actionText: '點擊刷新',

actionTextColor: '#fff',

onActionClick: function (e) {

location.reload()

},

})

} else {

var showBg =

document.documentElement.getAttribute('data-theme') === 'light'

? '#49b1f5'

: '#1f1f1f'

var cssText = `top: 0; background: ${showBg};`

document.getElementById('app-refresh').style.cssText = cssText

}

}

</script>

Html

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

<style type="text/css">

.app-refresh {

position: fixed;

top: -2.2rem;

left: 0;

right: 0;

z-index: 99999;

padding: 0 1rem;

font-size: 15px;

height: 2.2rem;

transition: all 0.3s ease;

}

.app-refresh-wrap {

display: flex;

color: #fff;

height: 100%;

align-items: center;

justify-content: center;

}

.app-refresh-wrap span {

color: #fff;

text-decoration: underline;

cursor: pointer;

}

</style>

運行

在你運行 hexo g 後，記得要運行 gulp 這樣才會生效

以下是結合了上面提到的 Gulp 壓縮和 PWA 的 gulpfile.js，可供參考。

Js

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

var gulp = require('gulp');

var cleanCSS = require('gulp-clean-css');

var htmlmin = require('gulp-htmlmin');

var htmlclean = require('gulp-htmlclean');

var imagemin = require('gulp-imagemin');

var workbox = require("workbox-build");

// babel

var uglify = require('gulp-uglify');

var babel = require('gulp-babel');

//tester

// var uglifyjs = require("terser");

// var composer = require('gulp-uglify/composer');

// var pump = require('pump');

// var minify = composer(uglifyjs, console);

//pwa

gulp.task('generate-service-worker', () => {

return workbox.injectManifest({

swSrc: './sw-template.js',

swDest: './public/sw.js',

globDirectory: './public',

globPatterns: [

"\*\*/\*.{html,css,js,json,woff2}"

],

modifyURLPrefix: {

"": "./"

}

});

});

//minify js babel

gulp.task('compress', () =>

gulp.src(['./public/\*\*/\*.js', '!./public/\*\*/\*.min.js'])

.pipe(babel({

presets: ['@babel/preset-env']

}))

.pipe(uglify().on('error', function(e){

console.log(e);

}))

.pipe(gulp.dest('./public'))

);

//minify js tester

// gulp.task('compress', function (cb) {

// var options = {};

// pump([

// gulp.src(['./public/\*\*/\*.js', '!./public/\*\*/\*.min.js']),

// minify(options),

// gulp.dest('./public')

// ],

// cb

// );

// });

//css

gulp.task('minify-css', () => {

return gulp.src('./public/\*\*/\*.css')

.pipe(cleanCSS({

compatibility: 'ie11'

}))

.pipe(gulp.dest('./public'));

});

// 壓縮 public 目錄內 html

gulp.task('minify-html', () => {

return gulp.src('./public/\*\*/\*.html')

.pipe(htmlclean())

.pipe(htmlmin({

removeComments: true, //清除 HTML 註釋

collapseWhitespace: true, //壓縮 HTML

collapseBooleanAttributes: true, //省略布爾屬性的值 <input checked="true"/> ==> <input />

removeEmptyAttributes: true, //刪除所有空格作屬性值 <input id="" /> ==> <input />

removeScriptTypeAttributes: true, //刪除 <script> 的 type="text/javascript"

removeStyleLinkTypeAttributes: true, //刪除 <style> 和 <link> 的 type="text/css"

minifyJS: true, //壓縮頁面 JS

minifyCSS: true, //壓縮頁面 CSS

minifyURLs: true

}))

.pipe(gulp.dest('./public'))

});

// 壓縮 public/uploads 目錄內圖片

gulp.task('minify-images', async () => {

gulp.src('./public/img/\*\*/\*.\*')

.pipe(imagemin({

optimizationLevel: 5, //類型：Number 預設：3 取值範圍：0-7（優化等級）

progressive: true, //類型：Boolean 預設：false 無失真壓縮jpg圖片

interlaced: false, //類型：Boolean 預設：false 隔行掃描gif進行渲染

multipass: false, //類型：Boolean 預設：false 多次優化svg直到完全優化

}))

.pipe(gulp.dest('./public/img'));

});

// 執行 gulp 命令時執行的任務

gulp.task("default", gulp.series("generate-service-worker", gulp.parallel(

'compress','minify-html', 'minify-css', 'minify-images'

)));

Icon

Butterfly 主題內置了 Font Awesome V5 圖標，目前已更新到 5.13.0，總共有 1,588 個免費圖標。由於是國外的圖標網站，對於國內的一些網站 Icon 並不支持。如有需要，你可以引入其它的圖標服務商。

iconfont

國內最出名的莫過於 iconfont, 功能很強大且圖標內容很豐富的矢量圖標庫。很多 Font Awesome 不支持的圖標都可以在這裏找到。同時，iconfont 支持選擇需要的圖標生成 css 鏈接，減少不必要的 CSS 加載。

註冊賬號

打開 iconfont 的網站，點擊導航欄的人像圖標，會跳出註冊界面，按要求註冊賬號。

添加圖標入庫

選擇自己需要的圖標，把鼠標移到圖標上，會顯示三個按鈕（依次是添加入庫、收藏和下載），而我們需要的是把圖標添加入庫

添加入庫後，你可以看到網站右上角購物車圖標顯示了 1 字，代表圖標已經添加入庫，點擊購物車圖標，會彈出側邊欄顯示詳情。

已選擇的圖標會顯示在上面，你可以重複上面的操作，把需要的圖標添加入庫，然後點擊添加到項目。

接下來會要求選擇項目名稱，沒有的自己創建一個。

生成 CSS 鏈接

在添加到項目之後，會跳到項目的詳情界面。點擊 Font class，然後再點擊暫無代碼，點擊生成文字。網站會自動生成 CSS 鏈接，我們只需要複製鏈接就行。

添加鏈接進 Butterfly.yml

打開 Butterfly.yml，找到 inject 配置，按要求把鏈接填入

在我們需要使用的地方填入 icon，例如 Menu，圖片使用格式為 iconfont icon名字

運行 Butterfly 之後，你就可以看到 menu 的圖標生效了

其他添加方法

除了通過引入 CSS 鏈接使用圖標，iconfont 也支持通過其它方法使用圖標，具體可查看 iconfont 官方使用文檔

其它圖標提供商

除了 iconfont，還有 RemixIcon、Flaticon 等等提供商，很多圖標可以選擇，具體使用方法請參考各自的文檔。

圖片壓縮

Butterfly 主題需要使用到很多圖片。如果圖片太大，會嚴重拖慢網站的加載速度。

圖片壓縮能夠有效的緩解這個問題。

除了通過 gulp-imagemin 來壓縮圖片，還可以通過在綫壓縮網站和軟件來進行壓縮。以下兩款是我自己正在使用的工具。網上有很多這樣的工具，挑選一款適合自己的就行。

tinypng

一個在綫壓縮的網站。壓縮後的圖片也保留了很高的質量，在知乎上很多人推薦，不過免費版有限制。

caesium

開源軟件，支持 Windows 和 macOS。可以批量壓縮軟件，無限制。

插件推薦

hexo-abbrlink

可以把鏈接 permalink 轉為數字的插件，配置容易，生成時自動轉為數字。

hexo-generator-feed

生成 RSS 文件的插件

hexo-filter-nofollow

為網站使用到的所有外鏈添加 rel="noopener external nofollow noreferrer", 可以有效地加強網站 SEO 和防止權重流失

hexo-generator-sitemap

生成 sitemap 的插件

hexo-generator-baidu-sitemap

看名字就知道，是專門為百度生成 sitemap 的插件

參考

利用 Workbox 實現博客的 PWA

漸進式網絡應用程式

作者: Jerry

連結: https://demo.jerryc.me/posts/4073eda/#%E6%8F%92%E4%BB%B6%E6%8E%A8%E8%96%A6

來源: Butterfly

著作權歸作者所有。商業轉載請聯絡作者獲得授權，非商業轉載請註明出處。

### 参考资源

https://github.com/jerryc127/hexo-theme-butterfly

https://www.antmoe.com/posts/a811d614/index.html

https://www.masantu.com/blog/2020-01-06/customize-hexo-theme-Butterfly/