VuePress 中文

书栈(BookStack.CN)

目　录

致谢

当前文档 《VuePress 中文&英文文档》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-07-15。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：http://www.bookstack.cn/books/VuePress-docs

书栈官网：http://www.bookstack.cn

书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

README

VuePress

> *Minimalistic docs generator with Vue component based layout system*

https://vuepress.vuejs.org/

Features

Built-in markdown extensions

- Table of Contents
- Custom Containers
- Line Highlighting
- Line Numbers
- Import Code Snippets

Using Vue in Markdown

- Templating
- Using Components

Vue-powered custom theme system

- Metadata
- Content Excerpt

Default theme

- Responsive layout
- Optional Homepage
- Simple out-of-the-box header-based search
- Algolia Search
- Customizable navbar and sidebar
- Auto-generated GitHub link and page edit links

Miscellaneous

- Multi-Language Support
- Service Worker
- Google Analytics
- Last Updated

Showcase

Websites built with VuePress:

Vue Ecosystem

- vue-cli
- vuex
- vue-server-renderer
- vue-router
- vue-test-utils
- vue-loader
- vetur
- rollup-plugin-vue

Community

- cr-vue
- vuesax

Todo Features

VuePress is still a work in progress. There are a few things that it currently does not support but are planned:

- Plugin support
- Blogging support

Contributions are welcome!

Development

> *Please make sure your version of Node.js is greater than 8.*

```
yarn
yarn dev  # serves VuePress' own docs with itself
yarn test # make sure your code change pass the test
```

License

MIT

来源(书栈小编注)

> *https://github.com/vuejs/vuepress*

中文文档

README

简洁至上

以 Markdown 为中心的项目结构，以最少的配置帮助你专注于写作。

Vue驱动

享受 Vue + webpack 的开发体验，可以在 Markdown 中使用 Vue 组件，又可以使用 Vue 来开发自定义主题。

高性能

details: VuePress 会为每个页面预渲染生成静态的 HTML，同时，每个页面被加载的时候，将作为 SPA 运行。

像数 1，2，3 一样容易

```
# 安装
yarn global add vuepress # 或者：npm install -g vuepress

# 新建一个 markdown 文件
echo '# Hello VuePress!' > README.md

# 开始写作
vuepress dev .

# 构建静态文件
vuepress build .
```

::: warning 注意
请确保你的 Node.js 版本 >= 8。
:::

介绍

介绍

VuePress 由两部分组成：一部分是支持用 Vue 开发主题的极简静态网站生成器，另一个部分是为书写技术文档而优化的默认主题。它的诞生初衷是为了支持 Vue 及其子项目的文档需求。

每一个由 VuePress 生成的页面都带有预渲染好的 HTML，也因此具有非常好的加载性能和搜索引擎优化（SEO）。同时，一旦页面被加载，Vue 将接管这些静态内容，并将其转换成一个完整的单页应用（SPA），其他的页面则会只在用户浏览到的时候才按需加载。

它是如何工作的？

事实上，一个 VuePress 网站是一个由 Vue、Vue Router 和 webpack 驱动的单页应用。如果你以前使用过 Vue 的话，当你在开发一个自定义主题的时候，你会感受到非常熟悉的开发体验，你甚至可以使用 Vue DevTools 去调试你的自定义主题。

在构建时，我们会为应用创建一个服务端渲染（SSR）的版本，然后通过虚拟访问每一条路径来渲染对应的HTML。这种做法的灵感来源于 Nuxt 的 `nuxt generate` 命令，以及其他的一些项目，比如 Gatsby。

特性

- 为技术文档而优化的 内置 Markdown 拓展
- 在 Markdown 文件中使用 Vue 组件的能力
- Vue 驱动的自定义主题系统
- 自动生成 Service Worker
- Google Analytics 集成
- 基于 Git 的“最后更新时间”
- 多语言支持
- 默认主题包含：
    - 响应式布局
    - 可选的主页
    - 简洁的开箱即用的标题搜索
    - Algolia 搜索
    - 可自定义的导航栏 和侧边栏
    - 自动生成的 GitHub 链接和页面的编辑链接

Todo

VuePress 仍然处于开发中，这里有一些目前还不支持、但已经在计划中的特性：

- 插件
- 博客系统

我们欢迎你为 VuePress 的开发作出贡献。

为什么不是...?

Nuxt

VuePress 能做的事情，Nuxt 理论上确实能够胜任，但 Nuxt 是为构建应用程序而生的，而 VuePress 则专注在以内容为中心的静态网站上，同时提供了一些为技术文档定制的开箱即用的特性。

Docsify / Docute

这两个项目同样都是基于 Vue，然而它们都是完全的运行时驱动，因此对 SEO 不够友好。如果你并不关注 SEO，同时也不想安装大量依赖，它们仍然是非常好的选择！

Hexo

Hexo 一直驱动着 Vue 的文档 —— 事实上，在把我们的主站从 Hexo 迁移到 VuePress 之前，我们可能还有很长的路要走。Hexo 最大的问题在于他的主题系统太过于静态以及过度地依赖纯字符串，而我们十分希望能够好好地利用 Vue 来处理我们的布局和交互，同时，Hexo 的 Markdown 渲染的配置也不是最灵活的。

GitBook

我们的子项目文档一直都在使用 GitBook。GitBook 最大的问题在于当文件很多时，每次编辑后的重新加载时间长得令人无法忍受。它的默认主题导航结构也比较有限制性，并且，主题系统也不是 Vue 驱动的。GitBook 背后的团队如今也更专注于将其打造为一个商业产品而不是开源工具。

快速上手

快速上手

::: warning 注意
请确保你的 Node.js 版本 >= 8。
:::

全局安装

如果你只是想尝试一下 VuePress，你可以全局安装它：

```
# 安装
yarn global add vuepress # 或者：npm install -g vuepress

# 新建一个 markdown 文件
echo "# Hello VuePress!" > README.md

# 开始写作
vuepress dev .

# 构建静态文件
vuepress build .
```

现有项目

如果你想在一个现有项目中使用 VuePress，同时想要在该项目中管理文档，则应该将 VuePress 安装为本地依赖。作为本地依赖安装让你可以使用持续集成工具，或者一些其他服务（比如 Netlify）来帮助你在每次提交代码时自动部署。

```
# 将 VuePress 作为一个本地依赖安装
yarn add -D vuepress # 或者：npm install -D vuepress

# 新建一个 docs 文件夹
mkdir docs

# 新建一个 markdown 文件
echo "# Hello VuePress!" > docs/README.md

# 开始写作
npx vuepress dev docs
```

::: warning
如果你的现有项目依赖了 webpack 3.x，推荐使用 Yarn 而不是 npm 来安装 VuePress。因为在这种情形下，

npm 会生成错误的依赖树。
:::

接着，在 `package.json` 里加一些脚本：

```
{
  "scripts": {
    "docs:dev": "vuepress dev docs",
    "docs:build": "vuepress build docs"
  }
}
```

然后就可以开始写作了：

```
yarn docs:dev # 或者：npm run docs:dev
```

要生成静态的 HTML 文件，运行：

```
yarn docs:build # 或者：npm run docs:build
```

默认情况下，文件将会被生成在 `.vuepress/dist`，当然，你也可以通过 `.vuepress/config.js` 中的 `dest` 字段来修改，生成的文件可以部署到任意的静态文件服务器上，参考 部署 来了解更多。

基本配置

基本配置

配置文件

如果没有任何配置，这个网站将会是非常局限的，用户也无法在你的网站上自由导航。为了更好地自定义你的网站，让我们首先在你的文档目录下创建一个 `.vuepress` 目录，所有 VuePress 相关的文件都将会被放在这里。你的项目结构可能是这样：

```
.
├─ docs
│  ├─ README.md
│  └─ .vuepress
│     └─ config.js
└─ package.json
```

一个 VuePress 网站必要的配置文件是 `.vuepress/config.js`，它应该导出一个 JavaScript 对象：

```
module.exports = {
  title: 'Hello VuePress',
  description: 'Just playing around'
}
```

对于上述的配置，如果你运行起 dev server，你应该能看到一个页面，它包含一个页头，里面包含一个标题和一个搜索框。VuePress 内置了基于 headers 的搜索 —— 它会自动为所有页面的标题、 `h2` 和 `h3` 构建起一个简单的搜索索引。

参见 配置 来查看所有可配置的选项。

::: tip 其他配置格式
你也可以使用 YAML (`.vuepress/config.yml`) 或是 TOML (`.vuepress/config.toml`) 格式的配置文件。
:::

主题配置

一个 VuePress 主题应该负责整个网站的布局和交互细节。在 VuePress 中，目前自带了一个默认的主题（正是你现在所看到的），它是为技术文档而设计的。同时，默认主题提供了一些选项，让你可以去自定义导航栏（navbar）、 侧边栏（sidebar）和 首页（homepage） 等，详情请参见 默认主题 。

如果你想开发一个自定义主题，可以参考 自定义主题。

应用级别的配置

由于 VuePress 是一个标准的 Vue 应用，你可以通过创建一个 `.vuepress/enhanceApp.js` 文件来做一些应用级别的配置，当该文件存在的时候，会被导入到应用内部。`enhanceApp.js` 应该 `export default` 一个钩子函数，并接受一个包含了一些应用级别属性的对象作为参数。你可以使用这个钩子来安装一些附加的 Vue 插件、注册全局组件，或者增加额外的路由钩子等：

```
export default ({
  Vue, // VuePress 正在使用的 Vue 构造函数
  options, // 附加到根实例的一些选项
  router, // 当前应用的路由实例
  siteData // 站点元数据
}) => {
  // ...做一些其他的应用级别的优化
}
```

静态资源

静态资源

相对路径

所有的 Markdown 文件都会被 webpack 编译成 Vue 组件，因此你可以，并且应该更倾向于使用相对路径（Relative URLs）来引用所有的静态资源：

```
1. ![An image](./image.png)
```

同样地，这在 `*.vue` 文件的模板中一样可以工作，图片将会被 `url-loader` 和 `file-loader` 处理，在运行生成静态文件的构建任务时，文件会被复制到正确的位置。

除此之外，你也使用 `~` 前缀来明确地指出这是一个 webpack 的模块请求，这将允许你通过 webpack 别名来引用文件或者 npm 的依赖：

```
1. ![Image from alias](~@alias/image.png)
2. ![Image from dependency](~some-dependency/image.png)
```

Webpack 的别名可以通过 `.vuepress/config.js` 中 `configureWebpack` 来配置，如：

```
1. module.exports = {
2.   configureWebpack: {
3.     resolve: {
4.       alias: {
5.         '@alias': 'path/to/some/dir'
6.       }
7.     }
8.   }
9. }
```

公共文件

有时，你可能需要提供一个静态资源，但是它们并不直接被你的任何一个 markdown 文件或者主题组件引用 —— 举例来说，favicons 和 PWA 的图标，在这种情形下，你可以将它们放在 `.vuepress/public` 中， 它们最终会被复制到生成的静态文件夹中。

基础路径

如果你的网站会被部署到一个非根路径，你将需要在 `.vuepress/config.js` 中设置 `base`，举例来说，如果你打算将你的网站部署到 `https://foo.github.io/bar/`，那么 `base` 的值就应该被设置为 `"/bar/"`（应当总是以斜

杠开始，并以斜杠结束)。

有了基础路径（Base URL），如果你希望引用一张放在 `.vuepress/public` 中的图片，你需要使用这样路径：`/bar/image.png`，然而，一旦某一天你决定去修改 `base`，这样的路径引用将会显得异常脆弱。为了解决这个问题，VuePress 提供了内置的一个 helper `$withBase`（它被注入到了 Vue 的原型上），可以帮助你生成正确的路径：

```
<img :src="$withBase('/foo.png')" alt="foo">
```

值得一提的是，你不仅可以在你的 Vue 组件中使用上述的语法，在 Markdown 文件中亦是如此。

最后补充一句，一个 `base` 路径一旦被设置，它将会自动地作为前缀插入到 `.vuepress/config.js` 中所有以 `/` 开始的资源路径中。

Markdown 拓展

Markdown 拓展

Header Anchors

所有的标题将会自动地应用 anchor 链接，anchor 的渲染可以通过 `markdown.anchor` 来配置。

链接

内部链接

内部的、并以 `.md` or `.html` 结尾的链接，将会被转换成 `<router-link>` 用于 SPA 导航。

站内的每一个子文件夹都应当有一个 `README.md` 文件，它会被自动编译为 `index.html`。

::: tip
在链接到一个文件夹的 `index.html` 时，确保你的链接以 `/` 结尾，否则该链接将导致 404。比如，用 `/config/` 而不是 `/config`。
:::

如果你想要链接到另一个 markdown 文件：

1. 确保链接以 `.html` 或 `.md` 结尾；
2. 确保路径大小写正确，因为路径的匹配是大小写敏感的。

示例

以如下的文件结构为例：

```
.
├─ README.md
├─ foo
│  ├─ README.md
│  ├─ one.md
│  └─ two.md
└─ bar
   ├─ README.md
   ├─ three.md
   └─ four.md
```

```
[Home](/) <!-- 跳转到根部的 README.md -->
[foo](/foo/) <!-- 跳转到 foo 文件夹的 index.html -->
[foo heading anchor](/foo/#heading) <!-- 跳转到 foo/index.html 的特定 anchor 位置 -->
```

```
4. [foo - one](/foo/one.html) <!-- 具体文件可以使用 .html 结尾 -->
5. [foo - two](/foo/two.md) <!-- 也可以用 .md -->
```

外部链接

外部的链接将会被自动地设置为 `target="_blank" rel="noopener noreferrer"` ：

- vuejs.org
- VuePress on GitHub

你可以自定义通过配置 config.markdown.externalLinks 来自定义外部链接的特性。

Front Matter

VuePress 提供了对 YAML front matter 开箱即用的支持：

```
1. ---
2. title: Blogging Like a Hacker
3. lang: en-US
4. ---
```

这些数据可以在当前页的正文中使用，在任意的自定义或主题组件中，它可以通过 `$page` 来访问。

`title` 和 `lang` 的 meta 将会被自动地注入到当前的页面中，当然你也可以指定一些额外需要注入的 meta：

```
1. ---
2. meta:
3.   - name: description
4.     content: hello
5.   - name: keywords
6.     content: super duper SEO
7. ---
```

其他格式的 Front Matter

除了 YAML 之外，VuePress 也支持 JSON 或者 TOML 格式的 front matter。

JSON front matter 需要以花括号开头和结尾：

```
1. ---
2. {
3.   "title": "Blogging Like a Hacker",
4.   "lang": "en-US"
5. }
6. ---
```

TOML front matter 需要显式地标注为 TOML：

```
---toml
title = "Blogging Like a Hacker"
lang = "en-US"
---
```

GitHub 风格的表格

Input

```
| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |
```

Output

Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

Emoji

Input

```
:tada: :100:
```

Output

目录

Input

```
[[toc]]
```

Output

[[toc]]

目录（Table of Contents）的渲染可以通过 `markdown.toc` 选项来配置。

自定义容器

Input

```
::: tip
This is a tip
:::

::: warning
This is a warning
:::

::: danger
This is a dangerous warning
:::
```

Output

::: tip
This is a tip
:::

::: warning
This is a warning
:::

::: danger
This is a dangerous thing
:::

你也可以自定义块中的标题:

```
::: danger STOP
Danger zone, do not proceed
:::
```

::: danger STOP
Danger zone, do not proceed
:::

代码块中的行高亮

Input

```
``` js{4}
export default {
  data () {
    return {
```
```

```
      msg: 'Highlighted!'
    }
  }
}
```

Output

```
export default {
  data () {
    return {
      msg: 'Highlighted!'
    }
  }
}
```

行号

你可以通过配置来为每个代码块显示行号：

```
module.exports = {
  markdown: {
    lineNumbers: true
  }
}
```

- 示例：

" class="reference-link">导入代码段

你可以通过下述的语法导入已经存在的文件中的代码段：

```
<<< @/filepath
```

它也支持 行高亮：

```
<<< @/filepath{highlightLines}
```

Input

```
<<< @/test/markdown/fragments/snippet.js{2}
```

Output

<<< @/test/markdown/fragments/snippet.js{2}

::: tip 注意
由于代码段的导入将在 webpack 编译之前执行，因此你无法使用 webpack 中的路径别名，此处的 `@` 默认值是 `process.cwd()` 。
:::

进阶配置

VuePress 使用 markdown-it 来渲染 Markdown，上述大多数的拓展也都是通过自定义的插件实现的。想要进一步的话，你可以通过 `.vuepress/config.js` 的 `markdown` 选项，来对当前的 `markdown-it` 实例做一些自定义的配置：

```
module.exports = {
  markdown: {
    // markdown-it-anchor 的选项
    anchor: { permalink: false },
    // markdown-it-toc 的选项
    toc: { includeLevel: [1, 2] },
    config: md => {
      // 使用更多的 markdown-it 插件!
      md.use(require('markdown-it-xxx'))
    }
  }
}
```

在 **Markdown** 中 使用 **Vue**

在 Markdown 中 使用 Vue

浏览器的 API 访问限制

当你在开发一个 VuePress 应用时，由于所有的页面在生成静态 HTML 时都需要通过 Node.js 服务端渲染，因此所有的 Vue 相关代码都应当遵循 编写通用代码 的要求。简而言之，请确保只在 `beforeMount` 或者 `mounted` 访问浏览器 / DOM 的 API。

如果你正在使用，或者需要展示一个对于 SSR 不怎么友好的组件（比如包含了自定义指令），你可以将它们包裹在内置的 `<ClientOnly>` 组件中：

```
<ClientOnly>
  <NonSSRFriendlyComponent/>
</ClientOnly>
```

请注意，这并不能解决一些组件或库在导入时就试图访问浏览器 API 的问题 —— 如果需要使用这样的组件或库，你需要在合适的生命周期钩子中动态导入它们：

```
<script>
export default {
  mounted () {
    import('./lib-that-access-window-on-import').then(module => {
      // use code
    })
  }
}
</script>
```

模板语法

插值

每一个 Markdown 文件将首先被编译成 HTML，接着作为一个 Vue 组件传入 `vue-loader`，这意味着你可以在文本中使用 Vue 风格的插值：

Input

```
{{ 1 + 1 }}
```

Output

```
{{ 1 + 1 }}
```

指令

同样地，也可以使用指令：

Input

```
<span v-for="i in 3">{{ i }} </span>
```

Output

```
{{ i }}
```

访问网站以及页面的数据

编译后的组件没有私有数据，但可以访问 网站的元数据，举例来说：

Input

```
{{ $page }}
```

Output

```
{
  "path": "/using-vue.html",
  "title": "Using Vue in Markdown",
  "frontmatter": {}
}
```

Escaping

默认情况下，块级 (block) 的代码块将会被自动包裹在 `v-pre` 中。如果你想要在内联 (inline) 的代码块或者普通文本中显示原始的大括号，或者一些 Vue 特定的语法，你需要使用自定义容器 `v-pre` 来包裹：

Input

```
::: v-pre
`{{ This will be displayed as-is }}`
:::
```

Output

::: v-pre
`{{ This will be displayed as-is }}`
:::

使用组件

所有在 `.vuepress/components` 中找到的 `*.vue` 文件将会自动地被注册为全局的异步组件，如：

```
.
└ .vuepress
   └ components
      ├ demo-1.vue
      ├ OtherComponent.vue
      └ Foo
         └ Bar.vue
```

你可以直接使用这些组件在任意的 Markdown 文件中（组件名是通过文件名取到的）：

```
<demo-1/>
<OtherComponent/>
<Foo-Bar/>
```

::: warning 重要！
请确保一个自定义组件的名字包含连接符或者是 PascalCase，否则，它将会被视为一个内联元素，并被包裹在一个 `<p>` 标签中，这将会导致 HTML 渲染紊乱，因为 HTML 标准规定， `<p>` 标签中不允许放置任何块级元素。
:::

使用预处理器

VuePress 对以下预处理器已经内置相关的 webpack 配置： `sass` 、 `scss` 、 `less` 、 `stylus` 和 `pug` 。要使用它们你只需要在项目中安装对应的依赖即可。例如，要使用 `sass` ，需要安装：

```
yarn add -D sass-loader node-sass
```

然后你就可以在 Markdown 或是组件中使用如下代码：

```
<style lang="sass">
  .title
    font-size: 20px
</style>
```

要在组件中使用 `<template lang="pug">` ，则需要安装 `pug` 和 `pug-plain-loader` ：

```
yarn add -D pug pug-plain-loader
```

::: tip
需要指出的是，如果你是一个 `stylus` 用户，你并不需要在你的项目中安装 `stylus` 和 `stylus-loader` ，因为 VuePress 已经内置了它们。

对于那些没有内置的预处理器，除了安装对应的依赖，你还需要 拓展内部的 Webpack 配置。

:::

脚本和样式提升

有时，你可以只想在当前页面应用一些 JavaScript 或者 CSS，在这种情况下，你可以直接在 Markdown 文件中使用原生的 `<script>` 或者 `<style>` 标签，它们将会从编译后的 HTML 文件中提取出来，并作为生成的 Vue 单文件组件的 `<script>` 和 `<style>` 标签。

内置的组件

" class="reference-link">OutboundLink

() 用来表明当前是一个外部链接。在 VuePress 中这个组件会紧跟在每一个外部链接后面。

" class="reference-link">ClientOnly

参考 浏览器的 API 访问限制。

" class="reference-link">Content

- **Props**:
    - `custom` - boolean
- 用法:

当前的 `.md` 文件渲染的内容，当你在使用 自定义布局 时，它将非常有用。

```
1. <Content/>
```

参考:

- 自定义主题 > 获取渲染内容

" class="reference-link">Badge

- **Props**:
    - `text` - string
    - `type` - string, 可选值: `"tip"|"warn"|"error"`, 默认值是: `"tip"`
    - `vertical` - string, 可选值: `"top"|"middle"`, 默认值是: `"top"`
- **Usage**:

你可以在标题文本的末尾，使用这个组件来为某些 API 添加一些状态:

```
### Badge <Badge text="beta" type="warn"/> <Badge text="0.10.1+"/>
```

自定义主题

自定义主题

::: tip 提示
主题组件受到同样的 浏览器的 API 访问限制。
:::

VuePress 使用单文件组件来构建自定义主题。想要开发一个自定义主题，首先在你的文档根目录新建一个 `.vuepress/theme` 文件夹，然后再创建一个 `Layout.vue` 文件：

```
.
└ .vuepress
   └ theme
      └ Layout.vue
```

从这里开始，就和开发一个平时的 Vue 应用一样了，如何组织你的主题完全取决于你。

网站和页面的元数据

`Layout` 组件将会对每一个文档目录下的 `.md` 执行一次，同时，整个网站以及特定页面的元数据将分别暴露为 `this.$site` 和 `this.$page` 属性，它们将会被注入到每一个当前应用的组件中。

这是你现在看到的这个网站的 `$site` 的值：

```
{
  "title": "VuePress",
  "description": "Vue 驱动的静态网站生成器",
  "base": "/",
  "pages": [
    {
      "lastUpdated": 1524027677000,
      "path": "/",
      "title": "VuePress",
      "frontmatter": {}
    },
    ...
  ]
}
```

`title`，`description` 和 `base` 会从 `.vuepress/config.js` 中对应的的字段复制过来，而 `pages` 是一个包含了每个页面元数据对象的数据，包括它的路径、页面标题（明确地通过 YAML front matter 指定，或者通过该页面的第一个标题取到），以及所有源文件中的 `YAML front matter` 的数据。

下面的这个对象是你正在看的这个页面的 `$page` 的值：

```
{
  "lastUpdated": 1524847549000,
  "path": "/custom-themes.html",
  "title": "自定义主题",
  "headers": [/* ... */],
  "frontmatter": {}
}
```

如果用户在 `.vuepress/config.js` 配置了 `themeConfig`，你将可以通过 `$site.themeConfig` 访问到它。如此一来，你可以通过它来对用户开放一些自定义主题的配置 —— 比如指定目录或者页面的顺序，你也可以结合 `$site.pages` 来动态地构建导航链接。

最后，别忘了，作为 Vue Router API 的一部分，`this.$route` 和 `this.$router` 同样可以使用。

::: tip 提示
`lastUpdated` 是这个文件最后一次 git 提交的 UNIX 时间戳，更多细节请参考：最后更新时间。
:::

内容摘抄

如果一个 markdown 文件中有一个 `<!-- more -->` 注释，则该注释之前的内容会被抓取并暴露在 `$page.excerpt` 属性中。如果你在开发一个博客主题，你可以用这个属性来渲染一个带摘抄的文章列表。

获取渲染内容

当前的 `.md` 文件渲染的内容，可以作为一个独特的全局组件 `<Content/>` 来使用，你可能想要它显示在页面中的某个地方。一个最简单的主题，可以是一个唯一的 `Layout.vue` 组件，并包含以下内容：

```
<template>
  <div class="theme-container">
    <Content/>
  </div>
</template>
```

应用配置

自定义主题也可以通过主题根目录下的 `enhanceApp.js` 文件来对 VuePress 应用进行拓展配置。这个文件应当 `export default` 一个钩子函数，并接受一个包含了一些应用级别属性的对象作为参数。你可以使用这个钩子来安装一些附加的 Vue 插件、注册全局组件，或者增加额外的路由钩子等：

```
export default ({
  Vue, // VuePress 正在使用的 Vue 构造函数
  options, // 附加到根实例的一些选项
  router, // 当前应用的路由实例
  siteData // 站点元数据
}) => {
```

```
  // ...做一些其他的应用级别的优化
}
```

使用来自 npm 的主题

主题可以以 Vue 单文件组件的格式，并以 `vuepress-theme-xxx` 的名称发布到 npm 上。

如果想使用一个来自 npm 的主题，你需要在 `.vuepress/config.js` 补充 `theme` 选项：

```
module.exports = {
  theme: 'awesome'
}
```

VuePress 将会尝试去加载并使用位于 `node_modules/vuepress-theme-awesome/Layout.vue` 的主题组件。

修改默认主题

你可以使用 `vuepress eject [targetDir]` 这个命令来将默认主题的源码复制到 `.vuepress/theme` 文件夹下，从而可以对默认主题进行任意的修改。需要注意的是一旦 eject，即使升级 VuePress 你也无法再获得 VuePress 对默认主题的更新。

多语言支持

多语言支持

站点多语言配置

要启用 VuePress 的多语言支持，首先需要使用如下的文件结构：

```
docs
├─ README.md
├─ foo.md
├─ nested
│  └─ README.md
└─ zh
   ├─ README.md
   ├─ foo.md
   └─ nested
      └─ README.md
```

然后，在 `.vuepress/config.js` 中提供 `locales` 选项：

```
module.exports = {
  locales: {
    // 键名是该语言所属的子路径
    // 作为特例，默认语言可以使用 '/' 作为其路径。
    '/': {
      lang: 'en-US', // 将会被设置为 <html> 的 lang 属性
      title: 'VuePress',
      description: 'Vue-powered Static Site Generator'
    },
    '/zh/': {
      lang: 'zh-CN',
      title: 'VuePress',
      description: 'Vue 驱动的静态网站生成器'
    }
  }
}
```

如果一个语言没有声明 `title` 或者 `description`，VuePress 将会尝试使用配置顶层的对应值。如果每个语言都声明了 `title` 和 `description`，则顶层的这两个值可以被省略。

默认主题多语言配置

默认主题也内置了多语言支持，可以通过 `themeConfig.locales` 来配置。该选项接受同样的 `{ path: config }` 格式的值。每个语言除了可以配置一些站点中用到的文字之外，还可以拥有自己的 导航栏 和 侧边栏 配置：

```
module.exports = {
  locales: { /* ... */ },
  themeConfig: {
    locales: {
      '/': {
        selectText: 'Languages',
        label: 'English',
        editLinkText: 'Edit this page on GitHub',
        algolia: {},
        nav: [
          { text: 'Nested', link: '/nested/' }
        ],
        sidebar: {
          '/': [/* ... */],
          '/nested/': [/* ... */]
        }
      },
      '/zh/': {
        // 多语言下拉菜单的标题
        selectText: '选择语言',
        // 该语言在下拉菜单中的标签
        label: '简体中文',
        // 编辑链接文字
        editLinkText: '在 GitHub 上编辑此页',
        // 当前 locale 的 algolia docsearch 选项
        algolia: {},
        nav: [
          { text: '嵌套', link: '/zh/nested/' }
        ],
        sidebar: {
          '/zh/': [/* ... */],
          '/zh/nested/': [/* ... */]
        }
      }
    }
  }
}
```

部署

部署

下述的指南基于以下条件：

- 文档放置在项目的 `docs` 目录中；
- 使用的是默认的构建输出位置；
- VuePress 以本地依赖的形式被安装到你的项目中，并且配置了如下的 npm scripts：

```
{
  "scripts": {
    "docs:build": "vuepress build docs"
  }
}
```

GitHub Pages

1. 在 `docs/.vuepress/config.js` 中设置正确的 `base`。

   如果你打算发布到 `https://<USERNAME>.github.io/`，则可以省略这一步，因为 `base` 默认即是 `"/"`。

   如果你打算发布到 `https://<USERNAME>.github.io/<REPO>/`（也就是说你的仓库在 `https://github.com/<USERNAME>/<REPO>`），则将 `base` 设置为 `"/<REPO>/"`。

2. 在你的项目中，创建一个如下的 `deploy.sh` 文件（请自行判断去掉高亮行的注释）：

```
#!/usr/bin/env sh

# 确保脚本抛出遇到的错误
set -e

# 生成静态文件
npm run docs:build

# 进入生成的文件夹
cd docs/.vuepress/dist

# 如果是发布到自定义域名
# echo 'www.example.com' > CNAME

git init
git add -A
git commit -m 'deploy'

# 如果发布到 https://<USERNAME>.github.io
# git push -f git@github.com:<USERNAME>/<USERNAME>.github.io.git master
```

```
# 如果发布到 https://<USERNAME>.github.io/<REPO>
# git push -f git@github.com:<USERNAME>/<REPO>.git master:gh-pages

cd -
```

::: tip
你可以在你的持续集成的设置中，设置在每次 push 代码时自动运行上述脚本。
:::

GitLab Pages and GitLab CI

1. 在 `docs/.vuepress/config.js` 中设置正确的 `base`。

   如果你打算发布到 `https://<USERNAME or GROUP>.gitlab.io/`，则可以省略这一步，因为 `base` 默认即是 `"/"`。

   如果你打算发布到 `https://<USERNAME or GROUP>.gitlab.io/<REPO>/`（也就是说你的仓库在 `https://gitlab.com/<USERNAME>/<REPO>`），则将 `base` 设置为 `"/<REPO>/"`。

2. 在 `.vuepress/config.js` 中将 `dest` 设置为 `public`。

3. 在你项目的根目录下创建一个名为 `.gitlab-ci.yml` 的文件，无论何时你提交了更改，它都会帮助你自动构建和部署：

```
image: node:9.11.1

pages:
 cache:
   paths:
   - node_modules/

 script:
 - npm install
 - npm run docs:build
 artifacts:
   paths:
   - public
 only:
 - master
```

Netlify

1. 在 Netlify 中，创建一个新的 Github 项目，使用以下设置：
   - **Build Command:** `npm run build-docs` 或者 `yarn build-docs`
   - **Publish directory:** `docs/.vuepress/dist`
2. 点击 deploy 按钮！

Google Firebase

1. 请确保你已经安装了 firebase-tools。
2. 在你项目的根目录下创建 `firebase.json` 和 `.firebaserc`，并包含以下内容：

`firebase.json`：

```
{
 "hosting": {
   "public": "./docs/.vuepress/dist",
   "ignore": []
 }
}
```

`.firebaserc`：

```
{
 "projects": {
   "default": "<YOUR_FIREBASE_ID>"
 }
}
```

1. 在执行了 `yarn docs:build` 或 `npm run docs:build` 后，使用 `firebase deploy` 指令来部署。

Surge

1. 首先，假设你已经安装了 surge；
2. 运行 `yarn docs:build` 或者 `npm run docs:build`；
3. 想要使用 surge 来部署，你可以运行： `surge docs/.vuepress/dist`；

你也可以通过 `surge docs/.vuepress/dist yourdomain.com` 来部署到 自定义域名。

Heroku

1. 首先安装 Heroku CLI；
2. 在这里 注册一个 Heroku 账号；
3. 运行 `heroku login` 并填写你的 Heroku 证书：

```
heroku login
```

4. 在你的项目根目录中，创建一个名为 `static.json` 的文件，并包含下述内容：

`static.json`：

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

这里是你项目的配置，请参考 heroku-buildpack-static 了解更多。

1. 配置 Heroku 的 git 远程仓库:

```
# 版本变化
git init
git add .
git commit -m "My site ready for deployment."

# 以指定的名称创建一个新的 heroku 应用
heroku apps:create example

# 为静态网站设置构建包
heroku buildpacks:set https://github.com/heroku/heroku-buildpack-static.git
```

1. 部署你的网站:

```
# 发布网站
git push heroku master

# 打开浏览器查看 Helku CI 的 dashboard
heroku open
```

默认主题

默认主题

::: tip 提示
本页所列的选项仅对默认主题生效。如果你在使用一个自定义主题，选项可能会有不同。
:::

首页

默认的主题提供了一个首页（Homepage）的布局（用于 这个网站的主页）。想要使用它，需要在你的根级 `README.md` 的 YAML front matter 指定 `home: true`。以下是这个网站实际使用的数据：

```
---
home: true
heroImage: /hero.png
actionText: 快速上手 →
actionLink: /zh/guide/
features:
- title: 简洁至上
  details: 以 Markdown 为中心的项目结构，以最少的配置帮助你专注于写作。
- title: Vue驱动
  details: 享受 Vue + webpack 的开发体验，在 Markdown 中使用 Vue 组件，同时可以使用 Vue 来开发自定义主题。
- title: 高性能
  details: VuePress 为每个页面预渲染生成静态的 HTML，同时在页面被加载的时候，将作为 SPA 运行。
footer: MIT Licensed | Copyright © 2018-present Evan You
---
```

任何 `YAML front matter` 之后额外的内容将会以普通的 markdown 被渲染，并插入到 `features` 的后面。

导航栏

导航栏可能包含你的页面标题、搜索框、 导航栏链接、多语言切换、仓库链接，它们均取决于你的配置。

导航栏链接

你可以通过 `themeConfig.nav` 增加一些导航栏链接：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    nav: [
      { text: 'Home', link: '/' },
      { text: 'Guide', link: '/guide/' },
      { text: 'External', link: 'https://google.com' },
```

```
    ]
  }
}
```

当你提供了一个 `items` 数组而不是一个单一的 `link` 时，它将显示为一个 `下拉列表` ：

```
module.exports = {
  themeConfig: {
    nav: [
      {
        text: 'Languages',
        items: [
          { text: 'Chinese', link: '/language/chinese' },
          { text: 'Japanese', link: '/language/japanese' }
        ]
      }
    ]
  }
}
```

此外，你还可以通过嵌套的 `items` 来在 `下拉列表` 中设置分组：

```
module.exports = {
  themeConfig: {
    nav: [
      {
        text: 'Languages',
        items: [
          { text: 'Group1', items: [/*  */] },
          { text: 'Group2', items: [/*  */] }
        ]
      }
    ]
  }
}
```

禁用导航栏

你可以使用 `themeConfig.navbar` 来禁用所有页面的导航栏：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    navbar: false
  }
}
```

你也可以通过 `YAML front matter` 来禁用某个指定页面的导航栏：

```
---
navbar: false
---
```

侧边栏

想要使 侧边栏（Sidebar）生效，需要配置 `themeConfig.sidebar` ，基本的配置，需要一个包含了多个链接的数组：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: [
      '/',
      '/page-a',
      ['/page-b', 'Explicit link text']
    ]
  }
}
```

你可以省略 `.md` 拓展名，同时以 `/` 结尾的路径将会被视为 `*/README.md` ，这个链接的文字将会被自动获取到（无论你是声明为页面的第一个 header，还是明确地在 `YAML front matter` 中指定页面的标题）。如果你想要显示地指定链接的文字，使用一个格式为 `[link, text]` 的数组。

嵌套的标题链接

默认情况下，侧边栏会自动地显示由当前页面的标题（headers）组成的链接，并按照页面本身的结构进行嵌套，你可以通过 `themeConfig.sidebarDepth` 来修改它的行为。默认的深度是 `1` ，它将提取到 `h2` 的标题，设置成 `0` 将会禁用标题（headers）链接，同时，最大的深度为 `2` ，它将同时提取 `h2` 和 `h3` 标题。

也可以使用 `YAML front matter` 来为某个页面重写此值：

```
---
sidebarDepth: 2
---
```

" class="reference-link">显示所有页面的标题链接

默认情况下，侧边栏只会显示由当前活动页面的标题（headers）组成的链接，你可以将 `themeConfig.displayAllHeaders` 设置为 `true` 来显示所有页面的标题链接：

```
module.exports = {
  themeConfig: {
    displayAllHeaders: true // 默认值：false
  }
}
```

活动的标题链接

默认情况下，当用户通过滚动查看页面的不同部分时，嵌套的标题链接和 URL 中的 Hash 值会实时更新，这个行为可以通过以下的配置来禁用：

```
module.exports = {
  themeConfig: {
    activeHeaderLinks: false, // 默认值：true
  }
}
```

::: tip
值得一提的是，当你禁用此选项时，此功能的相应脚本将不会被加载，这是我们性能优化的一个小点。
:::

侧边栏分组

你可以通过使用对象来将侧边栏划分成多个组：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: [
      {
        title: 'Group 1',
        collapsable: false,
        children: [
          '/'
        ]
      },
      {
        title: 'Group 2',
        children: [ /* ... */ ]
      }
    ]
  }
}
```

侧边栏的每个子组默认是可折叠的，你可以设置 `collapsable: false` 来让一个组永远都是展开状态。

多个侧边栏

如果你想为不同的页面组来显示不同的侧边栏，首先，将你的页面文件组织成下述的目录结构：

```
.
├─ README.md
├─ contact.md
├─ about.md
├─ foo/
```

```
| ├ README.md
| ├ one.md
| └ two.md
└ bar/
  ├ README.md
  ├ three.md
  └ four.md
```

接着，遵循以下的侧边栏配置：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: {
      '/foo/': [
        '',     /* /foo/ */
        'one',  /* /foo/one.html */
        'two'   /* /foo/two.html */
      ],

      '/bar/': [
        '',      /* /bar/ */
        'three', /* /bar/three.html */
        'four'   /* /bar/four.html */
      ],

      // fallback
      '/': [
        '',        /* / */
        'contact', /* /contact.html */
        'about'    /* /about.html */
      ]
    }
  }
}
```

::: warning
确保 fallback 侧边栏被最后定义。VuePress 会按顺序遍历侧边栏配置来寻找匹配的配置。
:::

自动生成侧栏

如果你希望自动生成一个仅仅包含了当前页面标题（headers）链接的侧边栏，你可以通过 `YAML front matter` 来实现：

```
---
sidebar: auto
---
```

你也可以通过配置来在所有页面中启用它：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: 'auto'
  }
}
```

在 多语言 模式下，你也可以将其应用到某一特定的语言下：

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
     '/zh/': {
       sidebar: 'auto'
     }
  }
}
```

禁用侧边栏

你可以通过 `YAML front matter` 来禁用指定页面的侧边栏：

```
---
sidebar: false
---
```

搜索框

内置搜索

你可以通过设置 `themeConfig.search: false` 来禁用默认的搜索框，或是通过 `themeConfig.searchMaxSuggestions` 来调整默认搜索框显示的搜索结果数量：

```
module.exports = {
  themeConfig: {
    search: false,
    searchMaxSuggestions: 10
  }
}
```

::: tip
内置搜索只会为页面的标题、 `h2` 和 `h3` 构建搜索索引，如果你需要全文搜索，你可以使用 Algolia 搜索。
:::

Algolia 搜索

你可以通过 `themeConfig.algolia` 选项来用 Algolia 搜索 替换内置的搜索框。要启用 Algolia 搜索，你需要至少提供 `apiKey` 和 `indexName`：

```
module.exports = {
  themeConfig: {
    algolia: {
      apiKey: '<API_KEY>',
      indexName: '<INDEX_NAME>'
    }
  }
}
```

::: warning 注意
不同于开箱即用的 内置搜索，Algolia 搜索 需要你在使用之前将你的网站提交给它们用于创建索引。
:::

更多选项请参考 Algolia DocSearch 的文档。

最后更新时间

你可以通过 `themeConfig.lastUpdated` 选项来获取每个文件最后一次 `git` 提交的 UNIX 时间戳(ms)，同时它将以合适的日期格式显示在每一页的底部：

```
module.exports = {
  themeConfig: {
    lastUpdated: 'Last Updated', // string | boolean
  }
}
```

请注意，`themeConfig.lastUpdated` 默认是关闭的，如果给定一个字符串，它将会作为前缀显示（默认值是：`Last Updated`）。

::: warning 使用须知
由于 `lastUpdated` 是基于 `git` 的，所以你只能在一个基于 `git` 的项目中启用它。
:::

上 / 下一篇链接

上一篇和下一篇文章的链接将会自动地根据当前页面的侧边栏的顺序来获取。你也可以使用 `YAML front matter` 来明确地重写或者禁用它：

```
---
prev: ./some-other-page
next: false
---
```

Git 仓库和编辑链接

当你提供了 `themeConfig.repo` 选项，将会自动在每个页面的导航栏生成生成一个 GitHub 链接，以及在页面的底部生成一个 `"Edit this page"` 链接。

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    // 假定是 GitHub. 同时也可以是一个完整的 GitLab URL
    repo: 'vuejs/vuepress',
    // 自定义仓库链接文字。默认从 `themeConfig.repo` 中自动推断为
    // "GitHub"/"GitLab"/"Bitbucket" 其中之一，或是 "Source"。
    repoLabel: '查看源码',

    // 以下为可选的编辑链接选项

    // 假如你的文档仓库和项目本身不在一个仓库：
    docsRepo: 'vuejs/vuepress',
    // 假如文档不是放在仓库的根目录下：
    docsDir: 'docs',
    // 假如文档放在一个特定的分支下：
    docsBranch: 'master',
    // 默认是 false, 设置为 true 来启用
    editLinks: true,
    // 默认为 "Edit this page"
    editLinkText: '帮助我们改善此页面！'
  }
}
```

你可以通过 `YAML front matter` 来禁用指定页面的编辑链接：

```
---
editLink: false
---
```

简单的 CSS 覆盖

如果你只是希望应用一些简单的 overrides 到默认主题的样式上，你可以创建一个 `.vuepress/override.styl` 文件，这是一个 Stylus 文件，但是你仍然可以使用普通的 CSS 语法。

这里有一些你可以调整的颜色变量：

```
// showing default values
$accentColor = #3eaf7c
$textColor = #2c3e50
$borderColor = #eaecef
$codeBgColor = #282c34
```

自定义页面类

有时候你可能需要为特定页面添加一个 CSS 类名，以方便针对该页面添加一些专门的 CSS。这种情况下你可以在该页面的 YAML front matter 中声明一个 `pageClass` ：

```
---
pageClass: custom-page-class
---
```

然后你就可以写专门针对该页面的 CSS 了：

```
/* .vuepress/override.styl */

.theme-container.custom-page-class {
  /* 特定页面的 CSS */
}
```

特定页面的自定义布局

默认情况下，每个 `*.md` 文件将会被渲染在一个 `<div class="page">` 容器中，同时还有侧边栏、自动生成的编辑链接，以及上 / 下一篇文章的链接。如果你想要使用一个完全自定义的组件来代替当前的页面（而只保留导航栏），你可以再次使用 `YAML front matter` 来指定这个组件。

```
---
layout: SpecialLayout
---
```

这将会为当前的页面渲染 `.vuepress/components/SpecialLayout.vue` 布局。

配置

配置

基本配置

base

- 类型：`string`
- 默认值：`/`

部署站点的基础路径，如果你想让你的网站部署到一个子路径下，你将需要设置它。如 Github pages，如果你想将你的网站部署到 `https://foo.github.io/bar/`，那么 `base` 应该被设置成 `"/bar/"`，它的值应当总是以斜杠开始，并以斜杠结束。

`base` 将会自动地作为前缀插入到所有以 `/` 开始的其他选项的链接中，所以你只需要指定一次。

参考：

- Base URL
- 部署指南 > Github Pages

title

- 类型：`string`
- 默认值：`undefined`

网站的标题，它将会被用作所有页面标题的前缀，同时，默认主题下，它将显示在导航栏（navbar）上。

description

- 类型：`string`
- 默认值：`undefined`

网站的描述，它将会以 `<meta>` 标签渲染到当前页面的 HTML 中。

head

- 类型：`Array`
- 默认值：`[]`

额外的需要被注入到当前页面的 HTML `<head>` 中的标签，每个标签都可以以 `[tagName, { attrName: attrValue }, innerHTML?]` 的格式指定，举个例子，增加一个自定义的 favicon：

```
module.exports = {
  head: [
    ['link', { rel: 'icon', href: '/logo.png' }]
  ]
}
```

host

- Type: `string`
- Default: `'0.0.0.0'`

指定用于 dev server 的主机名。

port

- 类型: `number`
- 默认值: `8080`

指定 dev server 的端口。

dest

- 类型: `string`
- 默认值: `.vuepress/dist`

指定 `vuepress build` 的输出目录。

ga

- 类型: `string`
- 默认值: `undefined`

提供一个 Google Analytics ID 来使 GA 生效。

::: tip 提示
请留意 GDPR (2018年欧盟数据保护规则改革), 在合适或者需要的情况下，考虑将 Google Analytics 设置为匿名化的 IP。
:::

serviceWorker

- 类型: `boolean`
- 默认值: `false`

如果设置成 `true`，VuePress 将会自动生成并且注册一个 service worker，它缓存了那些已访问过的页面的内容，用于离线访问（仅在生产环境生效）。

如果你正在开发一个自定义主题，`Layout.vue` 组件将会自动触发下述的事件：

- `sw-ready`
- `sw-cached`
- `sw-updated`
- `sw-offline`
- `sw-error`

::: tip PWA NOTES
`serviceWorker` 选项仅仅用来控制 service worker，为了让你的网站完全地兼容 PWA，你需要在 `.vuepress/public` 提供 Manifest 和 icons，更多细节，请参见 MDN docs about the Web App Manifest.
此外，只有您能够使用 SSL 部署您的站点时才能启用此功能，因为 service worker 只能在 HTTPs 的 URL 下注册。
:::

locales

- 类型：`{ [path: string]: Object }`
- 默认值：`undefined`

提供多语言支持的语言配置。具体细节请查看 多语言支持。

shouldPrefetch

- 类型：`Function`
- 默认值：`() => true`

一个函数，用来控制对于哪些文件，是需要生成 `<link rel="prefetch">` 资源提示的。请参考 shouldPrefetch。

主题

theme

- 类型：`string`
- 默认值：`undefined`

当你使用自定义主题的时候，需要指定它。当值为 `"foo"` 时，VuePress 将会尝试去加载位于 `node_modules/vuepress-theme-foo/Layout.vue` 的主题组件。

themeConfig

- 类型：`Object`
- 默认值：`{}`

为当前的主题提供一些配置，这些选项依赖于你正在使用的主题。

也可以参考：

- 默认主题。

Markdown

markdown.lineNumbers

- 类型： `boolean`
- 默认值： `undefined`

是否在每个代码块的左侧显示行号。

参考：

- 行号

markdown.anchor

- 类型： `Object`
- 默认值： `{ permalink: true, permalinkBefore: true, permalinkSymbol: '#' }`

`markdown-it-anchor` 的选项。

markdown.externalLinks

- Type: `Object`
- Default: `{ target: '_blank', rel: 'noopener noreferrer' }`

这个键值对将会作为特性被增加到是外部链接的 `<a>` 标签上，默认的选项将会在新窗口中打开一个该外部链接。

markdown.toc

- 类型： `Object`
- 默认值： `{ includeLevel: [2, 3] }`

`markdown-it-table-of-contents` 的选项。

markdown.config

- 类型： `Function`
- 默认值： `undefined`

一个用于修改当前的 `markdown-it` 实例的默认配置，或者应用额外的插件的函数，举例如下：

```
module.exports = {
  markdown: {
    config: md => {
```

```
      md.set({ breaks: true })
      md.use(require('markdown-it-xxx'))
    }
  }
}
```

构建流程

postcss

- 类型：`Object`
- 默认值：`{ plugins: [require('autoprefixer')] }`

postcss-loader 的选项，请注意，指定这个值，将会覆盖内置的 autoprefixer，所以你需要自己将它加进去。

stylus

- Type: `Object`
- Default: `{ preferPathResolver: 'webpack' }`

stylus-loader 的选项。

scss

- Type: `Object`
- Default: `{}`

加载 `*.scss` 文件的 sass-loader 的选项。

sass

- Type: `Object`
- Default: `{ indentedSyntax: true }`

加载 `*.sass` 文件的 sass-loader 的选项。

less

- Type: `Object`
- Default: `{}`

less-loader 的选项。

configureWebpack

- 类型：`Object | Function`

- 默认值： `undefined`

用于修改内部的 Webpack 配置。如果给定一个对象，那么它将会被 webpack-merge 合并到最终的配置中，如果给定一个函数，它将会接受 `config` 作为第一个参数，以及 `isServer` 作为第二个参数，你可以直接更改 `config`，也可以返回一个待合并的对象。

```
module.exports = {
  configureWebpack: (config, isServer) => {
    if (!isServer) {
      // 修改客户端的 webpack 配置
    }
  }
}
```

chainWebpack

- 类型： `Function`
- 默认值： `undefined`

通过 webpack-chain 来修改内部的 Webpack 配置。

```
module.exports = {
  chainWebpack: (config, isServer) => {
    // config 是 ChainableConfig 的一个实例
  }
}
```

浏览器兼容性

evergreen

- 类型： `boolean`
- 默认值： `false`

如果你的对象只有那些“常青树”浏览器，你可以将其设置成 `true`，这将会禁止 ESNext 到 ES5 的转译以及对 IE 的 polyfills，同时会带来更快的构建速度和更小的文件体积。

英文文档

README

Simplicity First

Minimal setup with markdown-centered project structure helps you focus on writing.

Vue-Powered

Enjoy the dev experience of Vue + webpack, use Vue components in markdown, and develop custom themes with Vue.

Performant

VuePress generates pre-rendered static HTML for each page, and runs as an SPA once a page is loaded.

As Easy as 1, 2, 3

```
# install
yarn global add vuepress # OR npm install -g vuepress

# create a markdown file
echo '# Hello VuePress' > README.md

# start writing
vuepress dev
```

```
9.
10. # build to static files
11. vuepress build
```

::: warning COMPATIBILITY NOTE
VuePress requires Node.js >= 8.
:::

Introduction

Introduction

VuePress is composed of two parts: a minimalistic static site generator with a Vue-powered theming system, and a default theme optimized for writing technical documentation. It was created to support the documentation needs of Vue's own sub projects.

Each page generated by VuePress has its own pre-rendered static HTML, providing great loading performance and is SEO-friendly. Once the page is loaded, however, Vue takes over the static content and turns it into a full Single-Page Application (SPA). Additional pages are fetched on demand as the user navigates around the site.

How It Works

A VuePress site is in fact a SPA powered by Vue, Vue Router and webpack. If you've used Vue before, you will notice the familiar development experience when you are writing or developing custom themes (you can even use Vue DevTools to debug your custom theme!).

During the build, we create a server-rendered version of the app and render the corresponding HTML by virtually visiting each route. This approach is inspired by Nuxt's `nuxt generate` command and other projects like Gatsby.

Each markdown file is compiled into HTML with markdown-it and then processed as the template of a Vue component. This allows you to directly use Vue inside your markdown files and is great when you need to embed dynamic content.

Features

- Built-in markdown extensions optimized for technical documentation
- Ability to leverage Vue inside markdown files
- Vue-powered custom theme system
- Automatic Service Worker generation
- Google Analytics Integration
- "Last Updated" based on Git
- Multi-language support
- A default theme with:
    - Responsive layout
    - Optional Homepage
    - Simple out-of-the-box header-based search
    - Algolia Search

    - Customizable navbar and sidebar
    - Auto-generated GitHub link and page edit links

Todo

VuePress is still a work in progress. There are a few things that it currently does not support but are planned:

- Plugin support
- Blogging support

Contributions are welcome!

Why Not …?

Nuxt

Nuxt is capable of doing what VuePress does, but it is designed for building applications. VuePress is focused on content-centric static sites and provides features tailored for technical documentation out of the box.

Docsify / Docute

Both are great projects and also Vue-powered. Except they are both completely runtime-driven and therefore not SEO-friendly. If you don’t care about SEO and don’t want to mess with installing dependencies, these are still great choices.

Hexo

Hexo has been serving the Vue docs well - in fact, we are probably still a long way to go from migrating away from it for our main site. The biggest problem is that its theming system is very static and string-based - we really want to leverage Vue for both the layout and the interactivity. Also, Hexo’s markdown rendering isn’t the most flexible to configure.

GitBook

We’ve been using GitBook for most of our sub project docs. The primary problem with GitBook is that its development reload performance is intolerable with a large amount of files. The default theme also has a pretty limiting navigation structure, and the theming system is, again, not Vue based. The team behind GitBook is also more focused on turning it into a commercial product rather than an open-source tool.

Getting Started

Getting Started

::: warning COMPATIBILITY NOTE
VuePress requires Node.js >= 8.
:::

Global Installation

If you just want to play around with VuePress, you can install it globally:

```
# install globally
yarn global add vuepress # OR npm install -g vuepress

# create a markdown file
echo '# Hello VuePress' > README.md

# start writing
vuepress dev

# build
vuepress build
```

Inside an Existing Project

If you have an existing project and would like to keep documentation inside the project, you should install VuePress as a local dependency. This setup also allows you to use CI or services like Netlify for automatic deployment on push.

```
# install as a local dependency
yarn add -D vuepress # OR npm install -D vuepress

# create a docs directory
mkdir docs
# create a markdown file
echo '# Hello VuePress' > docs/README.md
```

::: warning
It is currently recommended to use Yarn instead of npm when installing VuePress into an existing project that has webpack 3.x as a dependency. Npm fails to generate the correct dependency tree in this case.
:::

Then, add some scripts to `package.json` :

```
{
  "scripts": {
    "docs:dev": "vuepress dev docs",
    "docs:build": "vuepress build docs"
  }
}
```

You can now start writing with:

```
yarn docs:dev # OR npm run docs:dev
```

To generate static assets, run:

```
yarn docs:build # Or npm run docs:build
```

By default the built files will be in `.vuepress/dist` , which can be configured via the `dest` field in `.vuepress/config.js` . The built files can be deployed to any static file server. See Deployment Guide for guides on deploying to popular services.

Configuration

Configuration

Config File

Without any configuration, the page is pretty minimal, and the user has no way to navigate around the site. To customize your site, let's first create a `.vuepress` directory inside your docs directory. This is where all VuePress-specific files will be placed in. Your project structure is probably like this:

```
.
├─ docs
│  ├─ README.md
│  └─ .vuepress
│     └─ config.js
└─ package.json
```

The essential file for configuring a VuePress site is `.vuepress/config.js`, which should export a JavaScript object:

```
module.exports = {
  title: 'Hello VuePress',
  description: 'Just playing around'
}
```

If you've got the dev server running, you should see the page now has a header with the title and a search box. VuePress comes with built-in headers-based search - it automatically builds a simple search index from the title, `h2` and `h3` headers from all the pages.

Consult the Config Reference for a full list of options.

::: tip Alternative Config Formats
You can also use YAML (`.vuepress/config.yml`) or TOML (`.vuepress/config.toml`) formats for the configuration file.
:::

Theme Configuration

A VuePress theme is responsible for all the layout and interactivity details of your site. VuePress ships with a default theme (you are looking at it right now) which is designed for technical documentation. It exposes a number of options that allow you to

customize the navbar, sidebar and homepage, etc. For details, check out the Default Theme Config page.

If you wish to develop a custom theme, see Custom Themes.

App Level Enhancements

Since the VuePress app is a standard Vue app, you can apply app-level enhancements by creating a file `.vuepress/enhanceApp.js`, which will be imported into the app if it is present. The file should `export default` a hook function which will receive an object containing some app level values. You can use this hook to install additional Vue plugins, register global components, or add additional router hooks:

```
export default ({
  Vue, // the version of Vue being used in the VuePress app
  options, // the options for the root Vue instance
  router, // the router instance for the app
  siteData // site metadata
}) => {
  // ...apply enhancements to the app
}
```

Asset Handling

Asset Handling

Relative URLs

All markdown files are compiled into Vue components and processed by webpack, therefore you can and **should prefer** referencing any asset using relative URLs:

```
1. ![An image](./image.png)
```

This would work the same way as in `*.vue` file templates. The image will be processed with `url-loader` and `file-loader`, and copied to appropriate locations in the generated static build.

In addition, you can use the `~` prefix to explicitly indicate this is a webpack module request, allowing you to reference files with webpack aliases or from npm dependencies:

```
1. ![Image from alias](~@alias/image.png)
2. ![Image from dependency](~some-dependency/image.png)
```

webpack aliases can be configured via configureWebpack in `.vuepress/config.js`. Example:

```
1. module.exports = {
2.   configureWebpack: {
3.     resolve: {
4.       alias: {
5.         '@alias': 'path/to/some/dir'
6.       }
7.     }
8.   }
9. }
```

Public Files

Sometimes you may need to provide static assets that are not directly referenced in any of your markdown or theme components - for example, favicons and PWA icons. In such cases you can put them inside `.vuepress/public` and they will be copied to the root of the generated directory.

Base URL

If your site is deployed to a non-root URL, you will need to set the `base` option in `.vuepress/config.js`. For example, if you plan to deploy your site to `https://foo.github.io/bar/`, then `base` should be set to `"/bar/"` (it should always start and end with a slash).

With a base URL, if you want to reference an image in `.vuepress/public`, you'd have to use URLs like `/bar/image.png`. However, this is brittle if you ever decide to change the `base` later. To help with that, VuePress provides a built-in helper `$withBase` (injected onto Vue's prototype) that generates the correct path:

```
<img :src="$withBase('/foo.png')" alt="foo">
```

Note you can use the above syntax not only in theme components, but in your markdown files as well.

In addition, if a `base` is set, it is automatically prepended to all asset URLs in `.vuepress/config.js` options.

Markdown Extensions

Markdown Extensions

Header Anchors

Headers automatically get anchor links applied. Rendering of anchors can be configured using the `markdown.anchor` option.

Links

Internal Links

Inbound links ending in `.md` or `.html` are converted to `<router-link>` for SPA navigation.

Each sub-directory in your static site should contain a `README.md`. It will automatically be converted to `index.html`.

::: tip
When writing the relative path to a directory's `index.html`, don't forget to close it off with a `/`, otherwise you will get a 404. For example, use `/config/` instead of `/config`.
:::

If you want to link to another markdown file within a directory, remember to:

1. Append it with either `.html` or `.md`
2. Make sure the case matches since the path is case-sensitive

Example

Given the following directory structure:

```
.
├─ README.md
├─ foo
│  ├─ README.md
│  ├─ one.md
│  └─ two.md
└─ bar
   ├─ README.md
   ├─ three.md
   └─ four.md
```

```
1. [Home](/) <!-- Sends the user to the root README.md -->
2. [foo](/foo/) <!-- Sends the user to index.html of directory foo -->
3. [foo heading anchor](/foo/#heading) <!-- Anchors user to a heading in the foo README file -->
4. [foo - one](/foo/one.html) <!-- You can append .html -->
5. [foo - two](/foo/two.md) <!-- Or you can append .md -->
```

External Links

Outbound links automatically gets `target="_blank" rel="noopener noreferrer"` :

- vuejs.org
- VuePress on GitHub

You can customize the attributes added to external links by setting `config.markdown.externalLinks`.

Front Matter

YAML front matter is supported out of the box:

```
1. ---
2. title: Blogging Like a Hacker
3. lang: en-US
4. ---
```

The data will be available to the rest of the page, plus all custom and theming components as `$page` .

`title` and `lang` will be automatically set on the current page. In addition you can specify extra meta tags to be injected:

```
1. ---
2. meta:
3.   - name: description
4.     content: hello
5.   - name: keywords
6.     content: super duper SEO
7. ---
```

Alternative Front Matter Formats

In addition, VuePress also supports JSON or TOML front matter.

JSON front matter needs to start and end in curly braces:

```
1. ---
2. {
```

```
  "title": "Blogging Like a Hacker",
  "lang": "en-US"
}
---
```

TOML front matter needs to be explicitly marked as TOML:

```
---toml
title = "Blogging Like a Hacker"
lang = "en-US"
---
```

GitHub-Style Tables

Input

```
| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |
```

Output

Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1

Emoji

Input

```
:tada: :100:
```

Output

A list of all emojis available can be found here.

Table of Contents

Input

```
1. [[toc]]
```

Output

[[toc]]

Rendering of TOC can be configured using the `markdown.toc` option.

Custom Containers

Input

```
1. ::: tip
2. This is a tip
3. :::
4.
5. ::: warning
6. This is a warning
7. :::
8.
9. ::: danger
10. This is a dangerous warning
11. :::
```

Output

::: tip
This is a tip
:::

::: warning
This is a warning
:::

::: danger
This is a dangerous thing
:::

You can also customize the title of the block:

```
1. ::: danger STOP
2. Danger zone, do not proceed
3. :::
```

::: danger STOP
Danger zone, do not proceed

:::

Line Highlighting in Code Blocks

Input

````
``` js{4}
export default {
  data () {
    return {
      msg: 'Highlighted!'
    }
  }
}
```
````

Output

```
export default {
  data () {
    return {
      msg: 'Highlighted!'
    }
  }
}
```

Line Numbers

You can enable line numbers for each code blocks via config:

```
module.exports = {
  markdown: {
    lineNumbers: true
  }
}
```

- Demo:

" class="reference-link">Import Code Snippets

You can import code snippets from existing files via following syntax:

```
<<< @/filepath
```

It also supports line highlighting:

```
<<< @/filepath{highlightLines}
```

Input

```
<<< @/test/markdown/fragments/snippet.js{2}
```

Output

<<< @/test/markdown/fragments/snippet.js{2}

::: tip
Since the import of the code snippets will be executed before webpack compilation, you can't use the path alias in webpack. The default value of `@` is `process.cwd()`.
:::

Advanced Configuration

VuePress uses markdown-it as the markdown renderer. A lot of the extensions above are implemented via custom plugins. You can further customize the `markdown-it` instance using the `markdown` option in `.vuepress/config.js`:

```
module.exports = {
  markdown: {
    // options for markdown-it-anchor
    anchor: { permalink: false },
    // options for markdown-it-toc
    toc: { includeLevel: [1, 2] },
    config: md => {
      // use more markdown-it plugins!
      md.use(require('markdown-it-xxx'))
    }
  }
}
```

Using Vue in Markdown

Using Vue in Markdown

Browser API Access Restrictions

Because VuePress applications are server-rendered in Node.js when generating static builds, any Vue usage must conform to the universal code requirements. In short, make sure to only access Browser / DOM APIs in `beforeMount` or `mounted` hooks.

If you are using or demoing components that are not SSR friendly (for example containing custom directives), you can wrap them inside the built-in `<ClientOnly>` component:

```
<ClientOnly>
  <NonSSRFriendlyComponent/>
</ClientOnly>
```

Note this does not fix components or libraries that access Browser APIs **on import** - in order to use code that assumes a browser environment on import, you need to dynamically import them in proper lifecycle hooks:

```
<script>
export default {
  mounted () {
    import('./lib-that-access-window-on-import').then(module => {
      // use code
    })
  }
}
</script>
```

Templating

Interpolation

Each markdown file is first compiled into HTML and then passed on as a Vue component to `vue-loader` . This means you can use Vue-style interpolation in text:

Input

```
{{ 1 + 1 }}
```

Output

```
{{ 1 + 1 }}
```

Directives

Directives also work:

Input

```
<span v-for="i in 3">{{ i }} </span>
```

Output

```
{{ i }}
```

Access to Site & Page Data

The compiled component does not have any private data but does have access to the site metadata. For example:

Input

```
{{ $page }}
```

Output

```
{
  "path": "/using-vue.html",
  "title": "Using Vue in Markdown",
  "frontmatter": {}
}
```

Escaping

By default, fenced code blocks are automatically wrapped with `v-pre`. If you want to display raw mustaches or Vue-specific syntax inside inline code snippets or plain text, you need to wrap a paragraph with the `v-pre` custom container:

Input

```
::: v-pre
`{{ This will be displayed as-is }}`
:::
```

Output

::: v-pre
`{{ This will be displayed as-is }}`
:::

Using Components

Any `*.vue` files found in `.vuepress/components` are automatically registered as global, async components. For example:

```
.
└─ .vuepress
   └─ components
      ├─ demo-1.vue
      ├─ OtherComponent.vue
      └─ Foo
         └─ Bar.vue
```

Inside any markdown file you can then directly use the components (names are inferred from filenames):

```
<demo-1/>
<OtherComponent/>
<Foo-Bar/>
```

::: warning IMPORTANT
Make sure a custom component's name either contains a hyphen or is in PascalCase. Otherwise it will be treated as an inline element and wrapped inside a `<p>` tag, which will lead to hydration mismatch because `<p>` does not allow block elements to be placed inside it.
:::

Using Pre-processors

VuePress has built-in webpack config for the following pre-processors: `sass`, `scss`, `less`, `stylus` and `pug`. All you need to do is installing the corresponding dependencies. For example, to enable `sass`, install the following in your project:

```
yarn add -D sass-loader node-sass
```

Now you can use the following in markdown and theme components:

```
<style lang="sass">
.title
  font-size: 20px
</style>
```

Using `<template lang="pug">` requires installing `pug` and `pug-plain-loader`:

```
yarn add -D pug pug-plain-loader
```

::: tip
If you are a Stylus user, you don't need to install `stylus` and `stylus-loader` in your project because VuePress uses Stylus internally.

For pre-processors that do not have built-in webpack config support, you will need to extend the internal webpack config in addition to installing the necessary dependencies.
:::

Script & Style Hoisting

Sometimes you may need to apply some JavaScript or CSS only to the current page. In those cases you can directly write root-level `<script>` or `<style>` blocks in the markdown file, and they will be hoisted out of the compiled HTML and used as the `<script>` and `<style>` blocks for the resulting Vue single-file component.

Built-In Components

" class="reference-link">OutboundLink

It() is used to indicate that this is an external link. In VuePress this component have been followed by every external link.

" class="reference-link">ClientOnly

See Browser API Access Restrictions.

" class="reference-link">Content

- **Props**:
  - `custom` - boolean
- **Usage**:

The compiled content of the current `.md` file being rendered. This will be very useful when you use Custom Layout.

```
<Content/>
```

Also see:

- Custom Themes > Content Outlet

" class="reference-link">Badge

- **Props**:
  - `text` - string
  - `type` - string, optional value: `"tip"|"warn"|"error"`, defaults to `"tip"`.
  - `vertical` - string, optional value: `"top"|"middle"`, defaults to `"top"`.
- **Usage**:

You can use this component at the end of header text to add some status for some API:

```
1. ### Badge <Badge text="beta" type="warn"/> <Badge text="0.10.1+"/>
```

Custom Themes

Custom Themes

::: tip
Theme components are subject to the same browser API access restrictions.
:::

VuePress uses Vue single file components for custom themes. To use a custom layout, create a `.vuepress/theme` directory in your docs root, and then create a `Layout.vue` file:

```
.
└ .vuepress
   └ theme
      └ Layout.vue
```

From there it's the same as developing a normal Vue application. It is entirely up to you how to organize your theme.

Site and Page Metadata

The `Layout` component will be invoked once for every `.md` file in `docs`, and the metadata for the entire site and that specific page will be exposed respectively as `this.$site` and `this.$page` properties which are injected into every component in the app.

This is the value of `$site` of this very website:

```
{
  "title": "VuePress",
  "description": "Vue-powered Static Site Generator",
  "base": "/",
  "pages": [
    {
      "lastUpdated": 1524027677000,
      "path": "/",
      "title": "VuePress",
      "frontmatter": {}
    },
    ...
  ]
}
```

`title`, `description` and `base` are copied from respective fields in `.vuepress/config.js`.

`pages` contains an array of metadata objects for each page, including its path, page title (explicitly specified in YAML front matter or inferred from the first header on the page), and any YAML front matter data in that file.

This is the `$page` object for this page you are looking at:

```
{
  "lastUpdated": 1524847549000,
  "path": "/guide/custom-themes.html",
  "title": "Custom Themes",
  "headers": [/* ... */],
  "frontmatter": {}
}
```

If the user provided `themeConfig` in `.vuepress/config.js`, it will also be available as `$site.themeConfig`. You can use this to allow users to customize behavior of your theme - for example, specifying categories and page order. You can then use these data together with `$site.pages` to dynamically construct navigation links.

Finally, don't forget that `this.$route` and `this.$router` are also available as part of Vue Router's API.

::: tip
`lastUpdated` is the UNIX timestamp of this file's last git commit, for more details, refer to Last Updated.

:::

Content Excerpt

If a markdown file contains a `<!-- more -->` comment, any content above the comment will be extracted and exposed as `$page.excerpt`. If you are building custom theme for blogging, this data can be used to render a post list with excerpts.

Content Outlet

The compiled content of the current `.md` file being rendered will be available as a special `<Content/>` global component. You will need to render it somewhere in your layout in order to display the content of the page. The simplest theme can be just a single `Layout.vue` component with the following content:

```
<template>
  <div class="theme-container">
    <Content/>
  </div>
</template>
```

App Level Enhancements

Themes can enhance the Vue app that VuePress uses by exposing an `enhanceApp.js` file at the root of the theme. The file should `export default` a hook function which will receive an object containing some app level values. You can use this hook to install additional Vue plugins, register global components, or add additional router hooks:

```
export default ({
  Vue, // the version of Vue being used in the VuePress app
  options, // the options for the root Vue instance
  router, // the router instance for the app
  siteData // site metadata
}) => {
  // ...apply enhancements to the app
}
```

Using Theme from a Dependency

Themes can be published on npm in raw Vue SFC format as `vuepress-theme-xxx` .

To use a theme from an npm dependency, provide a `theme` option in `.vuepress/config.js` :

```
module.exports = {
  theme: 'awesome'
}
```

VuePress will attempt to locate and use `node_modules/vuepress-theme-awesome/Layout.vue` .

Customizing the Default Theme

The `vuepress eject [targetDir]` command will copy the default theme source code into `.vuepress/theme` to allow complete customization. Note, however, once you eject, you are on your own and won’t be receiving future updates or bug fixes to the default theme even if you upgrade VuePress.

Internationalization

Internationalization

Site Level i18n Config

To leverage multi-language support in VuePress, you first need to use the following file structure:

```
docs
├─ README.md
├─ foo.md
├─ nested
│  └─ README.md
└─ zh
   ├─ README.md
   ├─ foo.md
   └─ nested
      └─ README.md
```

Then, specify the `locales` option in `.vuepress/config.js`:

```
module.exports = {
  locales: {
    // The key is the path for the locale to be nested under.
    // As a special case, the default locale can use '/' as its path.
    '/': {
      lang: 'en-US', // this will be set as the lang attribute on <html>
      title: 'VuePress',
      description: 'Vue-powered Static Site Generator'
    },
    '/zh/': {
      lang: 'zh-CN',
      title: 'VuePress',
      description: 'Vue 驱动的静态网站生成器'
    }
  }
}
```

If a locale does not have `title` or `description` VuePress will fallback to the root level values. You can omit the root level `title` and `description` as long as they are provided in each locale.

Default Theme i18n Config

The default theme also has built-in i18n support via `themeConfig.locales`, using the same `{ path: config }` format. Each locale can have its own nav and sidebar config, in addition to a few other text values used across the site:

```
module.exports = {
  locales: { /* ... */ },
  themeConfig: {
    locales: {
      '/': {
        // text for the language dropdown
        selectText: 'Languages',
        // label for this locale in the language dropdown
        label: 'English',
        // text for the edit-on-github link
        editLinkText: 'Edit this page on GitHub',
        // algolia docsearch options for current locale
        algolia: {},
        nav: [
          { text: 'Nested', link: '/nested/' }
        ],
        sidebar: {
          '/': [/* ... */],
          '/nested/': [/* ... */]
        }
      },
      '/zh/': {
        selectText: '选择语言',
        label: '简体中文',
        editLinkText: '在 GitHub 上编辑此页',
        nav: [
          { text: '嵌套', link: '/zh/nested/' }
        ],
        algolia: {},
        sidebar: {
          '/zh/': [/* ... */],
          '/zh/nested/': [/* ... */]
        }
      }
    }
  }
}
```

Deploying

Deploying

The following guides are based on a few shared assumptions:

- You are placing your docs inside the `docs` directory of your project;
- You are using the default build output location (`.vuepress/dist`);
- VuePress is installed as a local dependency in your project, and you have setup the following npm scripts:

```
{
  "scripts": {
    "docs:build": "vuepress build docs"
  }
}
```

GitHub Pages

1. Set correct `base` in `docs/.vuepress/config.js`.

   If you are deploying to `https://<USERNAME>.github.io/`, you can omit `base` as it defaults to `"/"`.

   If you are deploying to `https://<USERNAME>.github.io/<REPO>/`, (i.e. your repository is at `https://github.com/<USERNAME>/<REPO>`), set `base` to `"/<REPO>/"`.

2. Inside your project, create `deploy.sh` with the following content (with highlighted lines uncommented appropriately) and run it to deploy:

```
#!/usr/bin/env sh

# abort on errors
set -e

# build
npm run docs:build

# navigate into the build output directory
cd docs/.vuepress/dist

# if you are deploying to a custom domain
# echo 'www.example.com' > CNAME

git init
git add -A
```

```
git commit -m 'deploy'

# if you are deploying to https://<USERNAME>.github.io
# git push -f git@github.com:<USERNAME>/<USERNAME>.github.io.git master

# if you are deploying to https://<USERNAME>.github.io/<REPO>
# git push -f git@github.com:<USERNAME>/<REPO>.git master:gh-pages

cd -
```

::: tip
You can also run the above script in your CI setup to enable automatic deployment on each push.
:::

GitLab Pages and GitLab CI

1. Set correct `base` in `docs/.vuepress/config.js`.

   If you are deploying to `https://<USERNAME or GROUP>.gitlab.io/`, you can omit `base` as it defaults to `"/"`.

   If you are deploying to `https://<USERNAME or GROUP>.gitlab.io/<REPO>/`, (i.e. your repository is at `https://gitlab.com/<USERNAME>/<REPO>`), set `base` to `"/<REPO>/"`.

2. Set `dest` in `.vuepress/config.js` to `public`.

3. Create a file called `.gitlab-ci.yml` in the root of your project with the content below. This will build and deploy your site whenever you make changes to your content.

```
image: node:9.11.1

pages:
  cache:
    paths:
    - node_modules/

  script:
  - npm install
  - npm run docs:build
  artifacts:
    paths:
    - public
  only:
  - master
```

Netlify

1. On Netlify, setup up a new project from GitHub with the following settings:
    - **Build Command:** `npm run docs:build` or `yarn docs:build`
    - **Publish directory:** `docs/.vuepress/dist`
2. Hit the deploy button!

Google Firebase

1. Make sure you have firebase-tools installed.
2. Create `firebase.json` and `.firebaserc` at the root of your project with the following content:

`firebase.json` :

```
{
 "hosting": {
   "public": "./docs/.vuepress/dist",
   "ignore": []
 }
}
```

`.firebaserc` :

```
{
 "projects": {
   "default": "<YOUR_FIREBASE_ID>"
 }
}
```

1. After running `yarn docs:build` or `npm run docs:build`, deploy with the command `firebase deploy`.

Surge

1. First install surge, if you haven't already.
2. Run `yarn docs:build` or `npm run docs:build` .
3. Deploy to surge, by typing `surge docs/.vuepress/dist` .

You can also deploy to a custom domain by adding `surge docs/.vuepress/dist yourdomain.com` .

Heroku

1. First install Heroku CLI.

2. Create a Heroku account here.
3. Run `heroku login` and fill in your Heroku credentials:

```
heroku login
```

4. Create a file called `static.json` in the root of your project with the content below:

`static.json` :

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

This is the configuration of your site. see more at heroku-buildpack-static.

1. Set up your Heroku git remote:

```
# version change
git init
git add .
git commit -m "My site ready for deployment."

# creates a new app with a specified name
heroku apps:create example

# set buildpack for static sites
heroku buildpacks:set https://github.com/heroku/heroku-buildpack-static.git
```

1. Deploying Your Site

```
# publish site
git push heroku master

# opens a browser to view the Dashboard version of Heroku CI
heroku open
```

Default Theme Config

Default Theme Config

::: tip
All options listed on this page apply to the default theme only. If you are using a custom theme, the options may be different.
:::

Homepage

The default theme provides a homepage layout (which is used on the homepage of this very website). To use it, specify `home: true` plus some other metadata in your root `README.md`'s YAML front matter. This is the actual data used on this site:

```yaml
---
home: true
heroImage: /hero.png
actionText: Get Started →
actionLink: /guide/
features:
- title: Simplicity First
  details: Minimal setup with markdown-centered project structure helps you focus on writing.
- title: Vue-Powered
  details: Enjoy the dev experience of Vue + webpack, use Vue components in markdown, and develop custom themes with Vue.
- title: Performant
  details: VuePress generates pre-rendered static HTML for each page, and runs as an SPA once a page is loaded.
footer: MIT Licensed | Copyright © 2018-present Evan You
---
```

Any additional content after the `YAML front matter` will be parsed as normal markdown and rendered after the features section.

If you want to use a completely custom homepage layout, you can also use a Custom Layout.

Navbar

The Navbar may contain your page title, Search Box, Navbar Links, Languages and Repository Link, all of them depends on your configuration.

Navbar Links

You can add links to the navbar via `themeConfig.nav`:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    nav: [
      { text: 'Home', link: '/' },
      { text: 'Guide', link: '/guide/' },
      { text: 'External', link: 'https://google.com' },
    ]
  }
}
```

These links can also be dropdown menus if you provide an array of `items` instead of a `link`:

```
module.exports = {
  themeConfig: {
    nav: [
      {
        text: 'Languages',
        items: [
          { text: 'Chinese', link: '/language/chinese' },
          { text: 'Japanese', link: '/language/japanese' }
        ]
      }
    ]
  }
}
```

In addition, you can have sub groups inside a dropdown by having nested items:

```
module.exports = {
  themeConfig: {
    nav: [
      {
        text: 'Languages',
        items: [
          { text: 'Group1', items: [/*  */] },
          { text: 'Group2', items: [/*  */] }
        ]
      }
    ]
  }
}
```

Disable the Navbar

To disable the navbar globally, use `themeConfig.navbar`:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    navbar: false
  }
}
```

You can disable the navbar for a specific page via `YAML front matter`:

```
---
navbar: false
---
```

Sidebar

To enable the sidebar, use `themeConfig.sidebar`. The basic configuration expects an Array of links:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: [
      '/',
      '/page-a',
      ['/page-b', 'Explicit link text']
    ]
  }
}
```

You can omit the `.md` extension, and paths ending with `/` are inferred as `*/README.md`. The text for the link is automatically inferred (either from the first header in the page or explicit title in `YAML front matter`). If you wish to explicitly specify the link text, use an Array in form of `[link, text]`.

Nested Header Links

The sidebar automatically displays links for headers in the current active page, nested under the link for the page itself. You can customize this behavior using `themeConfig.sidebarDepth`. The default depth is `1`, which extracts the `h2` headers. Setting it to `0` disables the header links, and the max value is `2` which extracts both `h2` and `h3` headers.

A page can also override this value in using `YAML front matter`:

```
---
sidebarDepth: 2
---
```

" class="reference-link">Displaying Header Links of All Pages

The sidebar only displays links for headers in the current active page. You can display all header links for every page with `themeConfig.displayAllHeaders: true` :

```
module.exports = {
  themeConfig: {
    displayAllHeaders: true // Default: false
  }
}
```

Active Header Links

By default, the nested header links and the hash in the URL are updated as the user scrolls to view the different sections of the page. This behavior can be disabled with the following theme config:

```
module.exports = {
  themeConfig: {
    activeHeaderLinks: false, // Default: true
  }
}
```

::: tip
It is worth mentioning that when you disable this option, the corresponding script of this functionality will not be loaded. This is a small point in our performance optimization.
:::

Sidebar Groups

You can divide sidebar links into multiple groups by using objects:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: [
      {
        title: 'Group 1',
        collapsable: false,
        children: [
          '/'
        ]
      },
      {
```

```
        title: 'Group 2',
        children: [ /* ... */ ]
      }
    ]
  }
}
```

Sidebar groups are collapsable by default. You can force a group to be always open with `collapsable: false` .

Multiple Sidebars

If you wish to display different sidebars for different sections of content, first organize your pages into directories for each desired section:

```
.
├─ README.md
├─ contact.md
├─ about.md
├─ foo/
│  ├─ README.md
│  ├─ one.md
│  └─ two.md
└─ bar/
   ├─ README.md
   ├─ three.md
   └─ four.md
```

Then, update your configuration to define your sidebar for each section.

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: {
      '/foo/': [
        '',     /* /foo/ */
        'one',  /* /foo/one.html */
        'two'   /* /foo/two.html */
      ],

      '/bar/': [
        '',      /* /bar/ */
        'three', /* /bar/three.html */
        'four'   /* /bar/four.html */
      ],

      // fallback
      '/': [
        '',        /* / */
        'contact', /* /contact.html */
```

```
        'about'    /* /about.html */
      ]
    }
  }
}
```

::: warning
Make sure to define the fallback configuration last.

VuePress checks each sidebar config from top to bottom. If the fallback configuration was first, VuePress would incorrectly match `/foo/` or `/bar/four.html` because they both start with `/` .
:::

Auto Sidebar for Single Pages

If you wish to automatically generate a sidebar that contains only the header links for the current page, you can use `YAML front matter` on that page:

```
---
sidebar: auto
---
```

You can also enable it in all pages by using config:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    sidebar: 'auto'
  }
}
```

In multi-language mode, you can also apply it to a specific locale:

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    '/': {
      sidebar: 'auto'
    }
  }
}
```

Disabling the Sidebar

You can disable the sidebar on a specific page with `YAML front matter` :

```
---
```

```
2. sidebar: false
3. ---
```

Search Box

Built-in Search

You can disable the built-in search box with `themeConfig.search: false`, and customize how many suggestions to be shown with `themeConfig.searchMaxSuggestions`:

```
1. module.exports = {
2.   themeConfig: {
3.     search: false,
4.     searchMaxSuggestions: 10
5.   }
6. }
```

::: tip
Built-in Search only builds index from the title, `h2` and `h3` headers, if you need full text search, you can use Algolia Search.
:::

Algolia Search

The `themeConfig.algolia` option allows you to use Algolia DocSearch to replace the simple built-in search. To enable it, you need to provide at least `apiKey` and `indexName`:

```
1. module.exports = {
2.   themeConfig: {
3.     algolia: {
4.       apiKey: '<API_KEY>',
5.       indexName: '<INDEX_NAME>'
6.     }
7.   }
8. }
```

::: warning Note
Unlike the built-in search engine which works out of the box, Algolia DocSearch requires you to submit your site to them for indexing before it starts working.
:::

For more options, refer to Algolia DocSearch's documentation.

Last Updated

The `themeConfig.lastUpdated` option allows you to get the UNIX timestamp(ms) of each file's

last `git` commit, and it will also display at the bottom of each page with a appropriate format:

```
module.exports = {
  themeConfig: {
    lastUpdated: 'Last Updated', // string | boolean
  }
}
```

Note that it's `off` by default. If given `string`, it will be displayed as a prefix (default value: `Last Updated`).

::: warning
Since `lastUpdated` is based on `git`, so you can only use it in a `git` repository.
:::

Prev / Next Links

Prev and next links are automatically inferred based on the sidebar order of the active page. You can also explicitly overwrite or disable them using `YAML front matter`:

```
---
prev: ./some-other-page
next: false
---
```

Git Repo and Edit Links

Providing `themeConfig.repo` auto generates a GitHub link in the navbar and "Edit this page" links at the bottom of each page.

```
// .vuepress/config.js
module.exports = {
  themeConfig: {
    // Assumes GitHub. Can also be a full GitLab url.
    repo: 'vuejs/vuepress',
    // Customising the header label
    // Defaults to "GitHub"/"GitLab"/"Bitbucket" depending on `themeConfig.repo`
    repoLabel: 'Contribute!',

    // Optional options for generating "Edit this page" link

    // if your docs are in a different repo from your main project:
    docsRepo: 'vuejs/vuepress',
    // if your docs are not at the root of the repo:
    docsDir: 'docs',
    // if your docs are in a specific branch (defaults to 'master'):
    docsBranch: 'master',
```

```
    // defaults to false, set to true to enable
    editLinks: true,
    // custom text for edit link. Defaults to "Edit this page"
    editLinkText: 'Help us improve this page!'
  }
}
```

You can also hide the edit link on a specific page via `YAML front matter` :

```
---
editLink: false
---
```

Simple CSS Override

If you wish to apply simple overrides to the styling of the default theme, you can create an `.vuepress/override.styl` file. This is a Stylus file but you can use normal CSS syntax as well.

There are a few color variables you can tweak:

```
// showing default values
$accentColor = #3eaf7c
$textColor = #2c3e50
$borderColor = #eaecef
$codeBgColor = #282c34
```

" class="reference-link">Existing issues

In order to override the default variables mentioned above, `override.styl` will be imported at the end of the `config.styl` in default theme, and this file will be used by multiple files, so once you wrote styles here, your style will be duplicated by multiple times. See #637.

In fact, `style constants override` and `styles override` are two things, the former should be executed before any CSS is compiled, while the latter should be generated at the end of the CSS bundle, which has the highest priority.

" class="reference-link">Migrate your styles to `style.styl`

Start from `0.12.0` , we split `override.styl` into two APIs: `override.styl` and `style.styl` :

If you wrote styles at `override.styl` in the past, e.g.

```
// override.styl
$textColor = red // style constants override
```

```
3.
4. #my-style {} // styles override or custom styles.
```

You'll need to separate the style part to `style.styl` :

```
1. // override.styl
2. // SHOULD ONLY focus on style constants override.
3. $textColor = red
```

```
1. // style.styl
2. // SHOULD focus on styles override or your custom styles.
3. #my-style {}
```

Custom Page Class

Sometimes, you may need to add a unique class for a specific page so that you can target content on that page only in custom CSS. You can add a class to the theme container div with `pageClass` in `YAML front matter` :

```
1. ---
2. pageClass: custom-page-class
3. ---
```

Then you can write CSS targeting that page only:

```
1. /* .vuepress/override.styl */
2.
3. .theme-container.custom-page-class {
4.   /* page-specific rules */
5. }
```

Custom Layout for Specific Pages

By default the content of each `*.md` file is rendered in a `<div class="page">` container, along with the sidebar, auto-generated edit links and prev/next links. If you wish to use a completely custom component in place of the page (while only keeping the navbar), you can again specify the component to use using `YAML front matter` :

```
1. ---
2. layout: SpecialLayout
3. ---
```

This will render `.vuepress/components/SpecialLayout.vue` for the given page.

Ejecting

You can copy the default theme source code into `.vuepress/theme` to fully customize the theme using the `vuepress eject [targetDir]` command. Note, however, once you eject, you are on your own and won't be receiving future updates or bug fixes to the default theme even if you upgrade VuePress.

Config Reference

Config Reference

Basic Config

base

- Type: `string`
- Default: `/`

The base URL the site will be deployed at. You will need to set this if you plan to deploy your site under a sub path, for example GitHub pages. If you plan to deploy your site to `https://foo.github.io/bar/`, then `base` should be set to `"/bar/"`. It should always start and end with a slash.

The `base` is automatically prepended to all the URLs that start with `/` in other options, so you only need to specify it once.

Also see:

- Base URL
- Deploy Guide > Github Pages

title

- Type: `string`
- Default: `undefined`

Title for the site. This will be the prefix for all page titles, and displayed in the navbar in the default theme.

description

- Type: `string`
- Default: `undefined`

Description for the site. This will be rendered as a `<meta>` tag in the page HTML.

head

- Type: `Array`
- Default: `[]`

Extra tags to be injected to the page HTML `<head>`. Each tag can be specified in the form of `[tagName, { attrName: attrValue }, innerHTML?]`. For example, to add a custom favicon:

```
module.exports = {
  head: [
    ['link', { rel: 'icon', href: '/logo.png' }]
  ]
}
```

host

- Type: `string`
- Default: `'0.0.0.0'`

Specify the host to use for the dev server.

port

- Type: `number`
- Default: `8080`

Specify the port to use for the dev server.

dest

- Type: `string`
- Default: `.vuepress/dist`

Specify the output directory for `vuepress build`.

ga

- Type: `string`
- Default: `undefined`

Provide the Google Analytics ID to enable integration.

::: tip
Please be aware of GDPR (2018 reform of EU data protection rules) and consider setting Google Analytics to anonymize IPs where appropriate and/or needed.
:::

serviceWorker

- Type: `boolean`
- Default: `false`

If set to `true`, VuePress will automatically generate and register a service worker that caches the content for offline use (only enabled in production).

If developing a custom theme, the `Layout.vue` component will also be emitting the following events:

- `sw-ready`
- `sw-cached`
- `sw-updated`
- `sw-offline`
- `sw-error`

::: tip PWA NOTES
The `serviceWorker` option only handles the service worker. To make your site fully PWA-compliant, you will need to provide the Web App Manifest and icons in `.vuepress/public`. For more details, see MDN docs about the Web App Manifest.

Also, only enable this if you are able to deploy your site with SSL, since service worker can only be registered under HTTPs URLs.
:::

locales

- Type: `{ [path: string]: Object }`
- Default: `undefined`

Specify locales for i18n support. For more details, see the guide on Internationalization.

shouldPrefetch

- Type: `Function`
- Default: `() => true`

A function to control what files should have `<link rel="preload">` resource hints generated. See shouldPrefetch.

Theming

theme

- Type: `string`
- Default: `undefined`

Specify this to use a custom theme. With the value of `"foo"`, VuePress will attempt to load the theme component at `node_modules/vuepress-theme-foo/Layout.vue`.

themeConfig

- Type: `Object`
- Default: `{}`

Provide config options to the used theme. The options will vary depending on the theme you are using.

Also see:

- Default Theme Configuration.

Markdown

markdown.lineNumbers

- Type: `boolean`
- Default: `undefined`

Whether to show line numbers to the left of each code blocks.

Also see:

- Line Numbers

markdown.slugify

- Type: `Function`
- Default: source

Function for transforming header texts into slugs. This affects the ids/links generated for header anchors, table of contents and sidebar links.

markdown.externalLinks

- Type: `Object`
- Default: `{ target: '_blank', rel: 'noopener noreferrer' }`

The key and value pair will be added to `<a>` tags that points to an external link. The default option will open external links in a new window.

markdown.anchor

- Type: `Object`
- Default: `{ permalink: true, permalinkBefore: true, permalinkSymbol: '#' }`

Options for markdown-it-anchor. (Note: prefer `markdown.slugify` if you want to customize

header ids.)

markdown.toc

- Type: `Object`
- Default: `{ includeLevel: [2, 3] }`

Options for markdown-it-table-of-contents. (Note: prefer `markdown.slugify` if you want to customize header ids.)

markdown.config

- Type: `Function`
- Default: `undefined`

A function to modify default config or apply additional plugins to the markdown-it instance used to render source files. Example:

```
module.exports = {
  markdown: {
    config: md => {
      md.set({ breaks: true })
      md.use(require('markdown-it-xxx'))
    }
  }
}
```

Build Pipeline

postcss

- Type: `Object`
- Default: `{ plugins: [require('autoprefixer')] }`

Options for postcss-loader. Note specifying this value will overwrite autoprefixer and you will need to include it yourself.

stylus

- Type: `Object`
- Default: `{ preferPathResolver: 'webpack' }`

Options for stylus-loader.

scss

- Type: `Object`
- Default: `{}`

Options for sass-loader to load `*.scss` files.

sass

- Type: `Object`
- Default: `{ indentedSyntax: true }`

Options for sass-loader to load `*.sass` files.

less

- Type: `Object`
- Default: `{}`

Options for less-loader.

configureWebpack

- Type: `Object | Function`
- Default: `undefined`

Modify the internal webpack config. If the value is an Object, it will be merged into the final config using webpack-merge; If the value is a function, it will receive the config as the 1st argument and an `isServer` flag as the 2nd argument. You can either mutate the config directly, or return an object to be merged:

```
module.exports = {
  configureWebpack: (config, isServer) => {
    if (!isServer) {
      // mutate the config for client
    }
  }
}
```

chainWebpack

- Type: `Function`
- Default: `undefined`

Modify the internal webpack config with webpack-chain.

```
module.exports = {
  chainWebpack: (config, isServer) => {
    // config is an instance of ChainableConfig
  }
```

```
5. }
```

Browser Compatibility

evergreen

- Type: `boolean`
- Default: `false`

Set to `true` if you are only targeting evergreen browsers. This will disable ES5 transpilation and polyfills for IE, and result in faster builds and smaller files.

升级日志

0.12.0 (2018-07-12)

Bug Fixes

- **$build** npm audit vulnerability (close: #493)(#641) (8dde5d8)
- **$markdown**: wrong sidebar slugs and anchor link at content (close: #645) (c2eaff3)

Features

- **$core:**: version data layer (0c5b752)
- **$default-theme:** new file-level API: `style.styl`. (2f53f2f)
  i. Fixed overriding css variable doesn't work at `0.11.0` (close: #639)
  ii. Split `override.styl` into two APIs: `override.styl` and `style.styl`, the former will focus on ONLY the stylus constants override, while the latter will focus on styles override or custom styles. See also: https://vuepress.vuejs.org/default-theme-config/#simple-css-override.

0.11.0 (2018-07-08)

Bug Fixes

- **$default-theme:** indent-styled code is invisible (close: #609) (fd46a26)
- **$default-theme:** cannot get sidebar when sidebar config contains non-ASCII chars. (close: #628) (8837e7a)
- **$core:** override style issues (close: #637) (#638) (f998802)
  i. Duplicated generated `override style`.
  ii. Unexpected style order, `override style` should be at the end of the extracted style bundle. (ref: mini-css-extract-plugin#130)

Features

- **$default-theme:** page top slot (f4c1059)
- **$build:** set exitCode to non-zero when catching error (close: #598 & #570) (#615) (0907c7e)
- **$default-theme:** support display header links of all pages (close #534) (#595) (36bb6a4)

0.10.2 (2018-06-20)

Bug Fixes

- build cannot exit (close: #580) (fa473a7)
- duplicate description meta (close: #565) (de35315)
- edit page from Bitbucket (#569) (5479d6e)
- multiple markdown tokens in header text (#564) (ec330f0)
- setting HMR port (close: #582) (#586) (64bb80d)

Features

- refine Badge’s API (d68199d)

0.10.1 (2018-06-08)

Bug Fixes

- active side arrow not middle align (#508). (5fcac1b)
- **$default-theme:** code renders language css as c (close: #527) (777c4f1)
- **$default-theme:** table tag cannot scroll horizontally (close: #518) (#519) (e9cdee7)
- **$dev:** using config.yml/toml doesn’t reload changes (close: #520) (6048eb9)
- compilation error when chainWebpack’s code contains ! (close: #532) (3b5991f)
- reserve ‘*’ and ‘_’ when detecting escape char ‘\’ (close: #544). (4503cfc)
- search box throw a error with no suggestions (#510) (1186d6a)

Features

- **$seo:** show page title in front of site title (#522) (ffe12b9)
- add support to import files as code fence (#538) (26ecff7)
- better log (#506) (d53807e)
- enable header request Content-Range (#555) (825877c)
- headers badge (#540) (c3696d2)
- shouldPrefetch option for bundleRenderer (close: #463) (#514) (9cb174d)
- support “themeConfig.sidebar: ‘auto’” (close: #552) (56cbb5f)
- support generic markdown file path reference (#509) (292e4bc)

0.10.0 (2018-05-25)

Features

- upgrade to babel 7 + use @vue/babel-preset-app (c43c73d)

0.9.1 (2018-05-25)

Bug Fixes

- avoid cache error (close #492) (75cdc74)
- fix config reload cache busting (90f9689)
- lastUpdated looks bad when editLinks is false. (11b1830)
- wrong OutboundLink insertion position (close: #496) (af96f28)

Features

- allow for disabling of active hash on scroll (#489) (4c09627)
- support filename that contains non-ASCII and unicode chars (#473) (566e681)

0.9.0 (2018-05-22)

Bug Fixes

- $page is missing at 404 page (#388) (cefc8c3)
- avoid the searchbox exceeded out of screen in narrow screen (#254) (8f04081)
- code looks not good at small sreen (close: #350) (6514c8f)
- code looks not good at small sreen (close: #350) (d0ef06f)
- dropdown overlap due to word wrapping (close: #359) (#360) (c65a8b7)
- duplicate slash when docs dir is not set (#361) (0c59ed5)
- emoji doesn't work in toc (close: #417) (#418) (1b9012e)
- ensure `<script>` blocks in SFCs in node_modules are transpiled (4bf56d7)
- glob patterns error on windows (close: #348) (#400) (ab53998)
- highlight active link (#272) doesn't work with non-EN hash. (a51a31b)
- highlight line issue for empty lines (bc15841)
- highlight lines are cut when sliding (#437) (66bd797)
- image overflow at custom content (close: #381) (#383) (145cf4f)
- index file judgement bug (close: #306) (#308) (fefa16c)
- missing css source map at dev environment (#460) (d3025e5)
- missing title and desc in 404 and custom theme. (fcaee80)
- nav link highlight issue with i18n (close: #445) (596014f)
- postcss-loader warnings (close: #278) (34c7f99)
- potential duplicate iteration keys at dropdown (#249) (1417a35)
- relative link checking (31b8feb)
- remove style override limitation to custom theme (close: 404) (#405) (69bd59d)
- resolve custom theme from global cli (close: #392) (#399) (01142df)
- title cannot be number at front matter (#297) (5023d19)
- unexpected scroll behavior after clicking sidebar links (#298) (6081a3d)
- unexpected top blank space when navbar is disable (#316) (2bdc68e)
- unexpected warning when using non-ASCII chars as filename. (530912e)
- upgrade webpack-serve and avoid port conflict (close #424) (#425) (22ffe52)
- use v-for with key (#438) (2076f7b)

Features

- bump up webpack to 4.8.1 (close: #309) (9e3f005)
- code line numbers (close: #365) (#379) (9b42690)
- generate the timestamp of last updated for each doc (close #258) (#282) (d9b290b)
- handle telephone links (#325) (087467a)
- header extraction improvement (close: #238) (#271) (53c8489)
- hide edit link by page (close: #284) (#286) (d46819c)
- highlight current region in sidebar (#272) (6b6d268)
- last updated UI in default theme. (#338) (272df57)
- make code type insensitive (close: #347) (5e87b65)
- show OutboundLink icon for external links (#428) (942a2b9)
- support disable navbar globally (#246) (e725ad2)
- support global markdown config for attributes of external links (#358) (20e5bd8)
- support render $page.excerpt to HTML (close: #458) (9510b9f)
- support style lang postcss (close: #461) (881199a)
- using babel and support JSX in vue. (close: #318) (#336) (82cd8bd)

Performance Improvements

- vastly improve rebuild perf with caching (dfdc00c)

0.8.4 (2018-04-24)

Bug Fixes

- algolia regression - missing options (close #234) (b19bd89)

Features

- support disable navbar via front matter (close: #187) (#232) (504268c)

0.8.3 (2018-04-23)

Bug Fixes

- always write override.style (9861deb)

0.8.2 (2018-04-23)

Bug Fixes

- nav-item underline use $accentColor (#230) (ddb590d)

Features

- expose layout slots for injecting custom content (3814e88)

0.8.1 (2018-04-23)

Bug Fixes

- algolia regression (close #228) (44b1201)

0.8.0 (2018-04-23)

Bug Fixes

- algolia check should be checking themeConfig.algolia (504c21d)
- default to localhost on windows (close #221) (4d5c50e)
- fix emoji not showing on sidebars (#206) (bc2c83a)
- fix Sidebar link active logic (#215) (9c93d8f)
- Fix the style of repo link. (f55fa00)
- fix title inferrence regression (close #208) (52c20cf)
- renames index.js to enhanceApp.js (#226) (0170449)
- siteTitle vs pageTitle (cd9b788)

Features

- Add docsRepo (#155) (716aefe)
- add max search suggestions config (#163) (a16a5b4)
- Algolia DocSearch Integration (#201) (2f0da01)
- also expose siteData in enhanceApp.js (5157c6f)
- expose all css pre-processor's options. (close #169) (#178) (8f0755a)
- support built-in pug config and document using pro-processors at component (#151) (f322105)
- support excerpt extraction with `<!-- more -->` (close #174) (fa404dc)
- support for TOML front matter (#141) (#164) (70620ba)
- support toml config (#138) (d136e22)
- theme index enhancment support (#154) (d026801)

0.7.1 (2018-04-20)

Bug Fixes

- infer source link label from repo url (#168) (c1bbd05)
- Only add language dropdown when there has more than one locale configured. (#181)

(7f311da)
- prioritize frontmatter's title, description and lang (#180) (384c5c7), closes #177 #184
- redirect /foo to /foo/ during dev (close #183) (99bc0aa)
- show navbar in more conditions (close #170) (748fa7f)

0.7.0 (2018-04-18)

Bug Fixes

- disable typographer in markdown-it (close #139) (be42da5)
- ensure runnable when no locales are provided (a25d86c)
- fix yarn global install (fix #102) (1130318)
- handle links with encoded hash (f0a1a00)
- search for locales (4cf1232)

Features

- adjust i18n config + documentation (bccddbf)
- i18n for edit link text (6f5bac0)

0.6.1 (2018-04-18)

Bug Fixes

- handle headers that start with numbers (fix #121) (ad83169)
- make search locale-scoped (close #128) (846eb59)
- **nav:** unepxected error when themeConfig.nav isn't given. (close: #125) (#127) (f052472)
- service worker path (51c6eb2)
- use correct host in tip after the server has started (#130) (fd447ae)
- use header's slug as it is if possible (#119) (5f7e199)

Features

- enable source map in build error traces (efff472)
- **sidebar:** support click the part outside sidebar to close the sidebar. (#132) (c6c71af)

0.6.0 (2018-04-18)

Bug Fixes

- allow viewport scaling (close #110) (2b2a07d)
- cli build –dest flag (#97) (e32d90b)
- css safe (#96) (be82e09)
- **default-theme:** only show features div if provided (3f76bfe)
- ensure using the same markdown config when extracting headers (14d4d25)
- handle index.md when checking relative links (52d6672)

Features

- Multiple Language Support + Complete Chinese Translation (#48) (8bbc5f3)
- support yaml config (#115) (3088b3e)

0.5.1 (2018-04-17)

Bug Fixes

- correctly resolve not-found path (#90) (c3dd0b1)
- meta viewport for iOS tap delay (f95e245)
- support mailto links in NavLink + style tweaks (close #93) (62cd00e)
- upgrade webpack-chain, fix css optimization settings (close #91) (1bbfa43)

Features

- allow configuring host + default to 0.0.0.0 (close #86) (9936696)

0.5.0 (2018-04-16)

Features

- dropdown Items in Navbar (#13) (79f8f14)
- enhanceApp.js (#80) (37ea038)
- support adding custom page class in front matter (#85) (40ca73c), closes #84

0.4.2 (2018-04-16)

Bug Fixes

- proper minimum node version warning (eb07685)

0.4.1 (2018-04-16)

Bug Fixes

- always transpile lib directory (#73) (56e0392)
- avoid html-webpack-plugin requiring incomaptible webpack internals (4816bef)
- prioritize own deps + avoid serving wrong index.html (fix #69) (781e37a)
- redirect */index.html* to / (close #83) (52e04c4)
- remove override import when ejecting (close #56) (2d811ed)
- remove unnecessary spread (63816c1)

Features

- add styles (#60) (580774b)

0.4.0 (2018-04-15)

Features

- allow default theme to be copied as custom theme (98e1665)
- vuepress eject for customizing default theme (89538fa)

0.3.3 (2018-04-15)

Bug Fixes

- fix outbound nav links (close #37) (c909007)

0.3.2 (2018-04-15)

Bug Fixes

- added escaping of meta tag attribute value (#29) (15a1ac8)
- escape text in code block when lang is text or not specified #31 (#32) (ac4acab)
- **dev build:** use portfinder (#30) (f2a8229), closes #26
- generate better slugs for non latin langs (close #45) (e08e3d2)
- hoistedTags may not always be present (close #35) (ed33515)
- home link ‘/‘ shouldn’t always stays active (#47) (67c758e)
- images should have 100% max width (9e63974)
- renderChildren / sidebarDepth: 0 (42f63a8)

Features

- add warning about node version on startup (close #51) (1118327)

0.3.1 (2018-04-14)

Bug Fixes

- **style:** prevent scrollbar in code (#18) (a3db4d2)
- code margin on mobile (695440f)

Features

- commands now defaults targetDir to cwd. (#25) (22b7943), closes #8

0.3.0 (2018-04-14)

0.2.2 (2018-04-14)

0.2.1 (2018-04-14)

Bug Fixes

- fix vuepress cant resolve custom theme under .vuepress/theme (#3) (133bdb3)

0.2.0 (2018-04-13)

Features

- auto detect invalid inbound links (ca82906)
- google analytics (764ccd5)
- pwa (664a8e0)

0.1.0 (2018-04-12)

Bug Fixes

- css extraction (d293194)
- workaround for empty style chunk (701658a)

Features

- dev server (890f929)
- support nesting in sidebar (1964709)
- support style/script hoisting + css modules (f97e676)