# **[前端组件库文档解决方案](https://blog.csdn.net/w20101310/article/details/124192282?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-0-124192282-blog-115551772.pc_relevant_3mothn_strategy_and_data_recovery&spm=1001.2101.3001.4242.1&utm_rel)**

本篇主要分享什么内容：

常用的文档/静态站点生成工具有哪些

每个工具有什么特点

工具适应场景

前置概念

MarkDown

MDX MarkDown + JSX

YMAL Front Matter

组件库文档工具选型

AndD组件库文档是怎么制做的，使用了什么工具。

以AntD Button组件为例，我们看一下antd组件库的文档页面结构构成和文档生成：

Button文档

Button文档仓库源文件

网站文档仓库源文件

按钮类型

bisheng

AntD使用了bisheng来生成组件库文档，把MarkDown进行拼接和渲染成最终的文档展示页面。

静态站生成工具方案

vuePress

gitbook

MDX

MarkDown + JSX

支持导入React组件

支持remark 生态系统中的任何插件

Playground 实时修改，实时预览

基础

支持MarkDown语法

完全支持JSX 以<字符开头的行都视为JSX代码块

支持import 和 exports

import 组件 json数据 md或mdx文档

MDXProvider

提供MarkDown渲染HTML使用组件的映射 组件列表

Gatsby

Demo

初始化

npm init gatsby

1

npm install -g gatsby-cli

gatsby new

1

2

运行

npm run develop

1

特点：生态好，功能丰富，有各种各样的插件，支持MDX。

Gatsby 有一个强大的功能，称为数据层，使用 Gatsby 的数据层，您可以组合来自多个来源的数据，这让您可以为每种类型的数据选择最佳平台。

http://localhost:8000/\_\_\_graphql 中可以看到GraphQL数据

数据来源

Gatsby-source-\*

数据拉入：页面数据拉入 使用页面查询，页面中导出 query，通过graphql查询即可

​ 组件中拉入数据 可以使用useStaticQuery钩子拉入，

动态创建页面

Gatsby的 文件系统路由 API定义用于命名src/pages目录中文件的特殊语法，它允许您根据数据层中的节点集合为站点动态创建新页面。

JSDoc

根据javascript文件中注释信息，生成JavaScript应用程序或库、模块的API文档 的工具

安装

npm install -D jsdoc

1

使用

jsdoc xxx.js

1

默认会输出文档到out文件夹，可以通过--destination指定输出路径

jsdoc-to-markdown

TSDoc

https://tsdoc.org/play

React Styleguidist

StoryBook

​ 一个强大的集组件开发，查看，测试的文档工具，支持多种框架。使用”组件驱动开发“理念。

支持多种框架 React Vue Angular Ember Preact Svelte等

Tutorials

CDD

docsify

特点：

简单轻便

没有静态构建的 html 文件

多个主题

安装

npm i docsify-cli -g

1

初始化

docsify init ./docs

1

预览

docsify serve docs

1

目录结构

index.html 文件入口

README.md 主页

.nojekyll 防止GitHub Pages忽略以下划线开头的文件

侧边栏

创建 \_sidebar.md（支持目录层级嵌套）。

\_sidebar.md中页面会自动生成标题和子标题

自定义导航栏

html标签

\_navbar.md（同样支持目录层级嵌套，展示形式为弹窗）

封面

\_coverpage.md #/ 首页全屏展示

可以指定背景图和背景色

可以指定只展示封面

配置

window.$docsify = {

​ el:‘#app’, // 根元素

​ repo:‘docsifyjs/docsify/’, //Git仓库地址

​ maxLevel: 6, // 目录最大层级

​ loadNavbar: false, // 加载\_navbar.md作为导航栏(或者直接指定md路径)

​ loadSidebar: false, // 加载\_sidebar.md作为侧边栏

​ hideSidebar: true, // 隐藏侧边栏

​ subMaxLevel: 0, // 在自定义侧边栏中添加目录（最大层级）

​ auto2top: true, // 页面路径改变时滚动到屏幕顶部

​ homepage: ‘README.md’, // #/ 主页

​ basePath: ‘/path/’, // 基本路径, 可以将其设置为其他目录或其他域名

​ relativePath: false, // 如果为 true，则链接是相对于当前上下文的。

​ coverpage: false, // 封面 默认加载\_coverpage.md，也可以指定md路径

​ logo,

​ name,

​ nameLink,

​ markdown, // 自定义渲染MarkDown为HTML 文档

​ themeColor,

​ executeScript: true,

​ mergeNavbar: true, // 小屏幕上的导航栏将与侧边栏合并

​ externalLinkTarget: ‘\_self’, // default: ‘\_blank’ 打开默认连接方式

​ routerMode: ‘history’, // default: ‘hash’ 路由模式

​ onlyCover: false, // #/只展示封面

​ requestHeaders: { ‘x-token’: ‘xxx’, }, // 设置请求资源头

​ notFoundPage: true, // 加载\_404.md 或指定相应的md

​ vueComponents, // 注册vue组件, 可在md中直接使用

​ vueGlobalOptions,

​ vueMounts

}

主题 官方和社区制作的主题

插件 全文检索，谷歌分析，表情符号，第三方脚本支持，图片缩放，在github上编辑，jsfiddler Demo预览，复制到剪切板，Gitalk, 分页和标签等

PWA

SSR

嵌入文件：支持视频， 音频，iframe或代码块，甚至MarkDown

Docz

基于MDX进行了封装

完全使用Gatsby构建，可以使用Gatsby的插件和工具生态

零配置

TypeScript支持

安装

npm install docz # react react-dom

1

运行

"scripts": {

"docz:dev": "docz dev",

"docz:build": "docz build",

"docz:serve": "docz build && docz serve"

}

1

2

3

4

5

开发

创建.mdx文件即可(指定name和route)。

构建

npm run build # 生成静态资源在.docz/dist目录中

npm run build -- --dest docs-site-directory # 通过--dest 指定文档生成目录

1

2

也可以在配置中指定打包输出目录

// doczrc.js

export default {

dest: '/some-folder'

}

1

2

3

4

部署

构建之后可以使用任何静态站点托管服务进行部署。

MDX支持

可以直接引入.jsx/.tsx组件，样式；

内置组件

Playground

Playground支持编辑实时渲染，支持函数组件和State

Props

组件内的prop-types定义和typeScript的Interface会通过转换成表格展示

文档设置

使用YMAL自定义文档设置(也可以自定义属性，用于自定义theme)

---

name: My Document

route: /custom-route

menu: Documents

hidden: false

---

1

2

3

4

5

6

CSS预处理器

需要Gatsby提供的能力，安装插件

TypeScript支持

// doczrc.js

export default {

typescript: true

}

1

2

3

4

如果需要精确控制组件后缀，可以使用filterComponents and docgenConfig进行过滤

支持自定义主题

项目配置

基本配置

base 页面访问的basePath

src 指定组件存放目录

files 指定docz解析文件查找路径规则 默认会查找所有扩展名为.mdx的文件

ignore 需要忽略解析的文件

dest 指定docz build的目录

title Header展示title，默认会去package.json中name字段

description HTML中meta字段

typescript typescript支持 默认false .mdx文件中需要引入TypeScript组件则需要设置

propsParser props格式化 供渲染使用，禁用可以提升性能。

config 指定docz配置文件 默认顺序 docz.json | .doczrc | doczrc.json |doczrc.js | docz.config.js | docz.config.json

public 指定公共目录，绝对资源路径会从这个目录下取数据

editBranch 点击 Github 按钮时用于编辑文档的分支

host devServer地址 默认 ‘127.0.0.1’

port devServer 端口

构建流程

menu 可指定菜单中文档的顺序

plugins 指定要使用的插件数组

组件和HooksAPI

ComponentsProvider 将组件传递给 MDX，它们将在您将 Markdown 转换为 html 时使用

Playground 渲染组件并在其中显示代码的可编辑版本

Props 获取组件并根据组件中属性定义生成属性表的组件

useComponents 配合ComponentsProvider使用

useDocs 获取所有已解析文档的列表, 当要创建菜单或列表之类的内容时会很有用。

useMenus 返回 Docz 构建的菜单

useConfig 获取项目配置中项目配置对象

支持自定义插件和MDX插件

使用注意：每次涉及到路由的变化都需要重启生效，遇到缓存问题可以删除.docz文件夹后重启

// 一个简单的docz配置 doczrc.js

export default {

files: './docs/mdx/\*.{md,markdown,mdx}',

dest: './docs/site',

title: 'Flex-Ctrip-Offline',

typescript: true

}

1

2

3

4

5

6

7

Dumi

开箱即用

为组件开发而生，支持Markdown扩展，可以渲染组件

主题系统，支持自定义渲染样式

API自动生成，基于TypeScript类型定义自动生成组件API

组件开发脚手架

npx @umijs/create-dumi-lib # 初始化一个文档模式的组件库开发脚手架

npx @umijs/create-dumi-lib --site # 初始化一个站点模式的组件库开发脚手架 (比文档模式多一个主页，主页使用docs/index.md)

# 也可手动切换文档模式 => 站点模式: 修改.umirc.ts,添加mode:'site'

1

2

3

静态站点脚手架

npx @umijs/create-dumi-app

1

运行

npm install

npm start

1

2

构建及部署

npm run build

1

目录结构

├── README.md

├── docs # 组件库文档目录

│ ├── index.md # 组件库文档首页(不存在会使用README.md)

│ └── otherDir # 组件文档其他路由

│ ├── index.md

│ ├── sample.md

│ └── help.md

├── src # 组件库源码目录(单纯文档站点可忽略)

│ ├── Foo

│ └── index.ts

├── .umirc.ts # dumi配置文件

└── .fatherrc.ts # father-build的配置文件用于组件库打包

1

2

3

4

5

6

7

8

9

10

11

12

代码块

jsx和tsx的代码块会被dumi解析为React组件，并进行渲染。

dumi引入组件原则：

​ 像用户一样使用组件：直接引入组件库进行文档demo演示。不仅可以用来调试组件、编写文档，还能用来被用户直接拷贝到项目中使用。dumi会为我们自动创建组件库NPM包->组件库源代码的映射。

外部demo

可以引入外部文件作为demo渲染，并可支持查看demo源代码

<code src="/path/to/complex-demo.tsx"></code>

1

直接嵌入渲染

```jsx

/\*\*

\* inline: true

\*/

import React from 'react';

export default () => '我会被直接嵌入';

```

1

2

3

4

5

6

7

embed Markdow嵌套

<!-- 引入全量的 Markdown 文件内容 -->

<embed src="/path/to/some.md"></embed>

<!-- 根据行号引入指定行的 Markdown 文件内容 -->

<embed src="/path/to/some.md#L1"></embed>

<!-- 根据行号引入部分 Markdown 文件内容 -->

<embed src="/path/to/some.md#L1-L10"></embed>

<!-- 根据正则引入部分 Markdown 文件内容 -->

<embed src="/path/to/some.md#RE-/^[^\r\n]+/"></embed>

1

2

3

4

5

6

7

8

组件API自动生成

JS Doc注释 + TypeScript类型定义的方式实现组件API自动生成

如何在非-umi-项目中使用-dumi

DEMO理念

目前我们选择的是使用Docz来做为业务组件库的文档生成工具，下一篇会讲一下我们为什么选择Docz,它有什么优点。欢迎持续关注，微信公众号”混沌前端“

————————————————

版权声明：本文为CSDN博主「一颗小行星!」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/w20101310/article/details/124192282