文档系统介绍

本文档系统是基于 vuepress、vuepress-theme-hope完成的

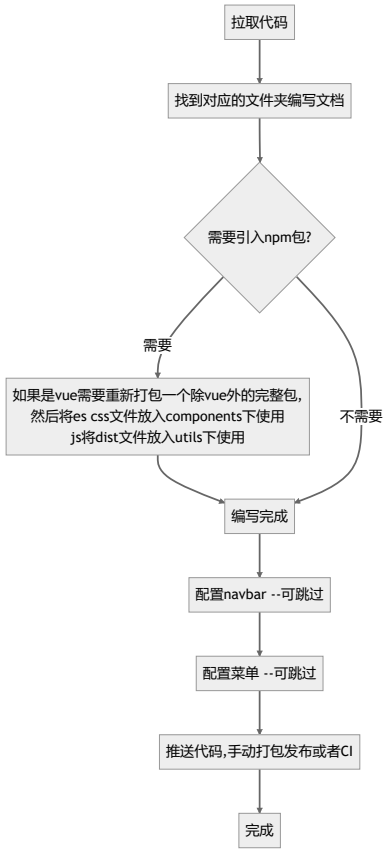
仓库地址

在线演示

目录结构

```
├── src
│   ├── cli → 存放cli相关的文档
│   ├── utils → 存放脚本工具相关的文档
│   ├── v2 → 存放Vue2组件的文档
│   ├── v3 → 存放Vue3组件的文档
│   ├── vscodeExtensions → 存放VSCode扩展的文档
│   ├── .vuepress → VuePress 配置文件夹
│   │   ├── dist → 构建输出目录
│   │   ├── components → 该文件夹下的组件会被自动注册，使用方式(<文件名 />，<文件夹名-文件名 />)
│   │   ├── navbar → 顶部导航栏配置文件
│   │   ├── public → 静态资源目录
│   │   │   ├── components → 用于存放vue3/vue2的npm组件，需要是es模式并且除vue外所有的依赖包需要打包进来。
│   │   │   ├──            → 目前不支持app.use()只能使用component的格式。
│   │   │   ├── utils → 工具脚本，可以在md文件中使用importmap引入
│   │   ├── styles → 用于存放样式相关的文件
│   │   ├── config.ts → 配置文件的入口文件
│   │   ├── client.ts → 客户端文件
│   │   ├── theme.ts → 主题配置文件
```

使用流程



使用方式

- 使用前可以先阅读注意事项，这里有一些模块和对应的代码可以参考。
- navbar 配置

> 配置文件地址: src\.vuepress\navbar内置 iconfont 图标

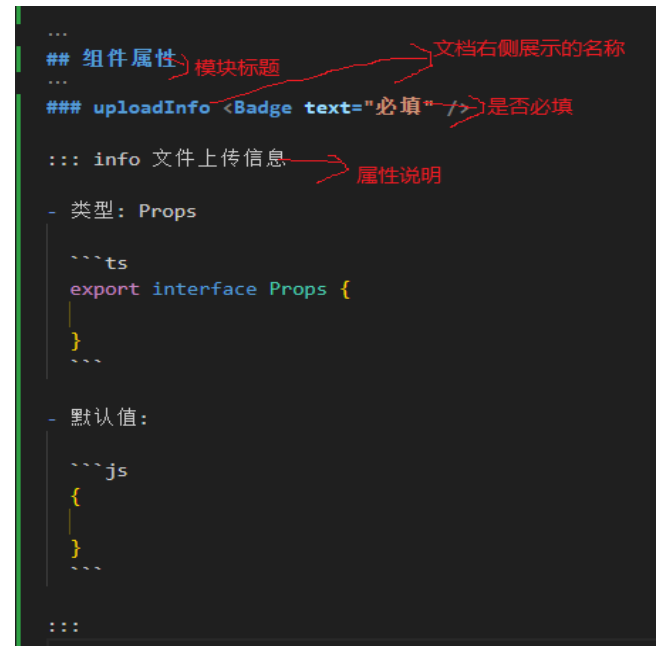
```
{
  "text": "navbar名称",
  "icon": "图标",
  "link": "/v3/(src/文件地址,不写具体md就会使用README.md当做首页)"
}
```

- 菜单配置

> vuepress-theme-hope 在 sidebar 中配置"structure"会根据目录结构自动生成菜单，该 navbar 下的 md 文件如果有文件夹会生成多层级的菜单，如果在最外层则生成一级菜单。

文档模块介绍

- 注意事项提供了几个模块可以复制对应的代码



Vue 组件代码演示

- theme-hope Vue交互演示地址
- 这个功能是 theme-hope 集成@vue/repl实现的
- 使用方式:
  1. 单独打包一份除了 vue 之外的完整包
  2. 将 es 文件和 css 复制到文档的 public/component 文件夹下使用
  3. 在 md 文件中的【imports】中引入
  4. 引入对应的样式和组件

@3

```
@import
\`\`\`json
{
  "imports": {
    "Com": "/components/v3/neo-custom-form-next/index.es.js",
    "utils": "/utils/index.js"
  }
}
\`\`\`
```

@4

```
<script setup>
import { ref } from "vue"; // vue是 @vue/repl 自带的 可以直接引用
import { CustomForm } from "Com"; // 引入组件 不支持插件
import { initStyle } from "utils";// 加载组件样式的脚本
initStyle("/components/v3/neo-custom-form-next/style.css"); // 加载组件样式
const list = ref([]);
const log = () => {
  console.log(list.value);
};
</script>

<template>
  <div>
    <button @click="log">获取数据</button>
    <CustomForm v-model:value="list" />
  </div>
</template>
```

脚本工具代码演示

- 使用方式:

1. 将dist文件复制到文档的 public/utils 文件夹下
2. 在md文件中的【imports】中引入
   - 如果工具引入了第三方包需要判断是什么类型的
     1. es文件 需要引入第三方包的es文件
     2. umd文件 需要引入第三方包带有全局变量的包,然后用initScript方法引入
3. 编写对应的案例代码

@2

```
@import
\`\`\`json
{
  "imports": {
    "moment": "https://unpkg.com/moment@2.29.4/src/moment.js",
    "momentTest": "/utils/momentTest/index.js"
  }
}
\`\`\`
```

@3

```
<script setup>
import {getNowDate} from 'momentTest'
const log = () => {
  console.log(getNowDate());
};
</script>

<template>
  <div @click="Log">获取当前日期
  </div>
</template>
```

文档痛点

1. 演示功能需要单独打包一个完整包复制到文档项目中
2. gitlab上面的npm包目前是tgz的格式，无法读取远程文件
3. 每个文档都需要单独去维护