

河南中医药大学信息技术学院（智能医疗行业学院）智能医学工程专业《互联网医疗服务开发》课程

第09章：View UI Plus与Vite

冯顺磊

河南中医药大学信息技术学院（智能医疗行业学院）
河南中医药大学信息技术学院智能医疗教研室
<https://aitcm.hactcm.edu.cn>
2025/12/3

本章概要

- View UI Plus
- Vite



1. View UI Plus

1.1 View UI Plus简介

- View UI Plus是一套基于Vue.js 3的高质量UI组件库，为开发者提供简洁、美观且功能丰富的用户界面组件，帮助开发者更高效地构建现代化的Web应用程序和管理系统。
- View UI Plus的优点
 - 组件丰富：提供大量高质量的组件，涵盖了从基础的按钮、输入框到复杂的表格、图表等各种类型。
 - 文档详尽：提供了全面的文档和丰富的示例，涵盖了组件的各个方面，包括属性、事件、插槽等。
 - 支持新特性：支持组合式API、Teleport等。
 - 按需引入：可根据实际引入组件，避免了引入整个组件库而导致的项目体积过大，有助于提高项目的加载速度和性能。

1. View UI Plus

1.2 基础组件

- Button按钮
 - Button按钮作为View UI Plus的基础组件，一般在触发业务逻辑时使用。
 - 按钮类型有：默认按钮、主按钮、虚线按钮、文字按钮以及四种颜色按钮，通过设置“type”为“primary”、“dashed”、“text”、“info”、“success”、“warning”、“error”创建不同样式的按钮，不设置为默认样式。

```
1. <template>
2.   <Button>Default</Button>
3.   <Button type="primary">Primary</Button>
4.   <Button type="dashed">Dashed</Button>
5.   <Button type="text">Text</Button>
6.   <br><br>
7.   <Button type="info">Info</Button>
8.   <Button type="success">Success</Button>
9.   <Button type="warning">Warning</Button>
10.  <Button type="error">Error</Button>
11.</template>
12.<script setup>
13.</script>
```



1. View UI Plus

1.2 基础组件

● Font 字体

- View UI Plus对CSS字体进行了统一规范，力求在不同平台、浏览器下能显示出其最佳的效果。

```
1. font-family: "Helvetica Neue", Helvetica, "PingFang SC", "Hiragino Sans GB", "Microsoft YaHei", "微软雅黑", Arial, sans-serif;
```

用数据说话	用数据说话	用数据说话
PingFang SC MAC OS 优先	Hiragino Sans GB 备用字体	Microsoft YaHei 次级备用字体
RGBa	RGBa	RGBa
Helvetica Neue 优先字体	Helvetica 备用字体	Arial 次级备用字体

1. View UI Plus

1.2 基础组件

● Color 色彩

- View UI Plus推荐使用以下调色板的颜色作为设计和开发规范，以保证页面和组件之间的视觉一致。

● 主色

- View UI Plus使用蓝色作为主色调，其中“Light Primary”常用于hover（鼠标经过），“Dark Primary”常用于active（选中）。其中Primary对应色值“#2d8cf0”，Light Primary对应色值“#5cadff”，Dark Primary对应色值“#2b85e4”。

Primary #2d8cf0	Light Primary #5cadff	Dark Primary #2b85e4
--------------------	--------------------------	-------------------------

● 辅助色

- 辅助色是具有代表性的颜色，常用于信息提示，比如成功、警告和失败。

Info #42a5f5	Success #4caf50	Warning #ff9800	Error #f44336
-----------------	--------------------	--------------------	------------------

● 中性色

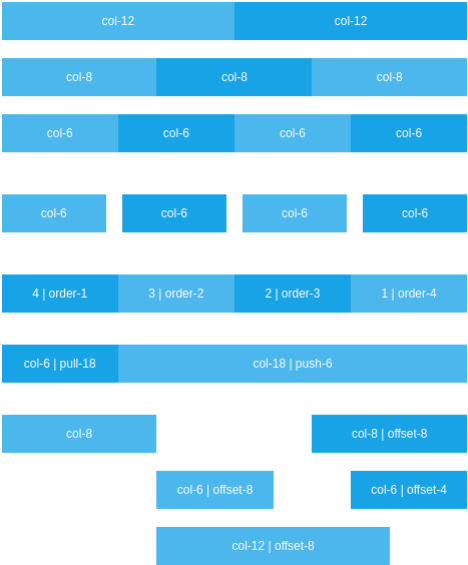
- 中性色常用于文本、背景、边框、阴影等，可以体现出页面的层次结构。

标题 Title #17233d	正文 Content #515a6e	辅助/图标 Sub Color #909895	失效 Disabled #bdbdbd
边框 Border #cccccc	分割线 Divider #cccccc	背景 Background #f5f5f5	

1. View UI Plus

1.3 布局组件

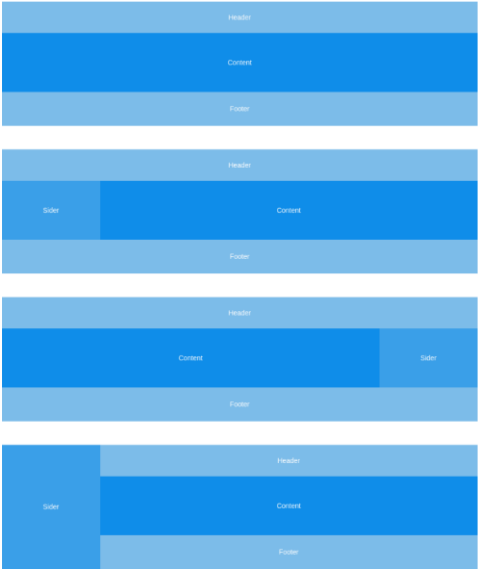
- Gird栅格
 - View UI Plus Gird组件采用了24栅格系统，将区域进行24等分。
 - View UI Plus Gird组件定义了行“row”和列“col”两个概念，具体使用方法如下。
 - 使用“row”在水平方向创建一行。
 - 将一组“col”插入在“row”中。
 - 在每个“col”中，键入自己的内容。
 - 通过设置“col”的“span”参数，指定跨越的范围，其范围是1到24。
 - 每个“row”中的“col”总和应该为24。



1. View UI Plus

1.3 布局组件

- Layout布局
 - Layout布局在View UI Plus组件中起到协助进行页面级整体布局的作用。
 - Layout: 布局容器，其下可嵌套“Header”、“Sider”、“Content”、“Footer”或Layout本身，可以放在任何父容器中。
 - Header: 顶部布局，自带默认样式，其下可嵌套任何元素，只能放在“Layout”中。
 - Sider: 侧边栏，自带默认样式及基本功能，其下可嵌套任何元素，只能放在“Layout”中。
 - Content: 内容部分，自带默认样式，其下可嵌套任何元素，只能放在“Layout”中。
 - Footer: 底部布局，自带默认样式，其下可嵌套任何元素，只能放在“Layout”中。



1. View UI Plus

1.4 导航组件

• Menu导航菜单

- Menu导航菜单是一个为页面和功能提供导航的菜单列表，常用于网站顶部和左侧。



1. View UI Plus

1.4 导航组件

• Tabs 标签页

- Tabs 标签页是一个选项卡切换组件，常用于平级区域大块内容的的收纳和展现。

```
1. <template>
2.   <Tabs>
3.     <TabPane label="标签一">标签一的内容</TabPane>
4.     <TabPane label="标签二" disabled>标签二的内容</TabPane>
5.     <TabPane label="标签三">标签三的内容</TabPane>
6.   </Tabs>
7. </template>
8. <script setup>
9. </script>
```



1. View UI Plus

1.3 组件样式

- Dropdown 下拉菜单
 - 完成基础的下拉菜单，需要配合 “DropdownMenu” 和 “DropdownItem” 两个组件来使用，并且给列表设置具名 “slot” 为 “list”。其中下拉菜单里触发的对象可以是链接、按钮等各种元素。

```
1. <template>
2.   <Dropdown>
3.     <a href="javascript:void(0)">
4.       下拉菜单
5.       <Icon type="ios-arrow-down">/Icon>
6.     </a>
7.     <DropdownMenu slot="list">
8.       <DropdownItem>驴打滚</DropdownItem>
9.       <DropdownItem>炸酱面</DropdownItem>
10.      <DropdownItem disabled>豆汁儿</DropdownItem>
11.      <DropdownItem>冰糖葫芦</DropdownItem>
12.      <DropdownItem divided>北京烤鸭</DropdownItem>
13.    </DropdownMenu>
14.  </Dropdown>
15.  <Dropdown style="margin-left: 20px">
16.    <Button type="primary">
17.      下拉菜单
18.      <Icon type="ios-arrow-down">/Icon>
19.    </Button>
20.    <DropdownMenu slot="list">
21.      <DropdownItem>驴打滚</DropdownItem>
22.      <DropdownItem>炸酱面</DropdownItem>
23.      <DropdownItem disabled>豆汁儿</DropdownItem>
24.      <DropdownItem>冰糖葫芦</DropdownItem>
25.      <DropdownItem divided>北京烤鸭</DropdownItem>
26.    </DropdownMenu>
27.  </Dropdown>
28. </template>
29. <script setup>
30. </script>
```



1. View UI Plus

1.4 导航组件

- Page 分页
 - 页面数据量较多时，可以使用Page 分页对数据进行切换。

```
1. <template>
2.   <Page :total="100" />
3. </template>
4. <script setup>
5. </script>
```



1.4 导航组件

- 面包屑主要使用于显示网站的层级结构，告知当前所在位置，以及向上级导航。

[Home](#) / [Components](#) / **Breadcrumb**

1.5 表单组件

- Input输入框是一个基本表单组件，支持“input”和“textarea”。

1. View UI Plus

1.5 表单组件

• Radio单选框

• Radio单选框主要用于一组可选项单项选择，或者单独用于切换到选中状态。

```
1. <template>
2.   <Radio v-model="single">Radio</Radio>
3. </template>
4. <script setup>
5.   import { ref } from 'vue';
6.   const single = ref(false);
7. </script>
```

☒ Apple ☐ Android ☐ Windows

☐ 金斑蝶 ☒ 爪哇犀牛 ☐ 印度黑羚

北京

上海

深圳

杭州

北京

上海

深圳

杭州

北京

上海

深圳

杭州

1. View UI Plus

1.5 表单组件

• Checkbox多选框

• Checkbox多选框主要用于一组可选项多项选择，或者单独用于标记切换某种状态。

```
1. <template>
2.   <Checkbox v-model="single">Checkbox</Checkbox>
3. </template>
4. <script setup>
5.   import { ref } from 'vue';
6.   const single = ref(false);
7. </script>
```

☐ Twitter ☒ Facebook ☒ Github ☐ Snapchat

☐ 香蕉 ☒ 苹果 ☐ 西瓜

全选

☒ 香蕉 ☐ 苹果 ☒ 西瓜

1. View UI Plus

1.5 表单组件

- Switch开关
 - Switch开关是在两种状态间切换时用到的开关选择器。

```
1. <template>
2.   <Switch v-model="switch1" @change="change" />
3. </template>
4. <script setup>
5.   import { ref } from 'vue';
6.   import { Message } from 'View UI Plus';
7.   const switch1 = ref(false);
8.   function change(status) {
9.     Message.info('开关状态: ' + status);
10.  }
11. </script>
```



1. View UI Plus

1.5 表单组件

- Table表格
 - Table表格是主要用于展示大量结构化数据的一个表单组件，它支持排序、筛选、分页、自定义操作、导出 csv 等复杂功能。

```
1. <template>
2.   <Table :columns="columns1" :data="data1"></Table>
3. </template>
4. <script setup>
5.   import { ref } from 'vue';
6.   const columns1 = ref([
7.     {
8.       title: 'Name',
9.       key: 'name'
10.    },
11.    {
12.      title: 'Age',
13.      key: 'age'
14.    },
15.    {
16.      title: 'Address',
17.      key: 'address'
18.    }
19.  ]);
20. const data1 = ref([
21.    {
22.      name: 'John Brown',
23.      age: 18,
24.      address: 'New York No. 1 Lake Park',
25.      date: '2016-10-03'
26.    },
27.    {
28.      name: 'Jim Green',
29.      age: 24,
30.      address: 'London No. 1 Lake Park',
31.      date: '2016-10-01'
32.    },
33.    {
34.      name: 'Joe Black',
35.      age: 30,
36.      address: 'Sydney No. 1 Lake Park',
37.      date: '2016-10-02'
38.    },
39.    {
40.      name: 'Jon Snow',
41.      age: 26,
42.      address: 'Ottawa No. 2 Lake Park',
43.      date: '2016-10-04'
44.    }
45.  ]);
46. </script>
```

1. View UI Plus

1.5 表单组件

- Table表格
 - Table表格是主要用于展示大量结构化数据的一个表单组件，它支持排序、筛选、分页、自定义操作、导出 csv 等复杂功能。

Name	Age	Address
John Brown	18	New York No. 1 Lake Park
Jim Green	24	London No. 1 Lake Park
Joe Black	30	Sydney No. 1 Lake Park
Jon Snow	26	Ottawa No. 2 Lake Park

Name	Age	Address
John Brown	18	New York No. 1 Lake Park
Jim Green	24	London No. 1 Lake Park
Joe Black	30	Sydney No. 1 Lake Park
Jon Snow	26	Ottawa No. 2 Lake Park

☐	Name	Age	Address
☐	John Brown	18	New York No. 1 Lake Park
☐	Jim Green	24	London No. 1 Lake Park
☐	Joe Black	30	Sydney No. 1 Lake Park
☐	Jon Snow	26	Ottawa No. 2 Lake Park

批量全选

取消全选

Date	Name	Age	Address
2015-10-03	John Brown	18	New York No. 1 Lake Park
2015-10-01	Jim Green	24	London No. 1 Lake Park
2016-10-02	Joe Black	30	Sydney No. 1 Lake Park
2015-10-04	Jon Snow	26	Ottawa No. 2 Lake Park

1. View UI Plus

1.5 表单组件

- Select选择器

```
1. <template>
2.   <Select v-model="model1" style="width:200px">
3.     <Option v-for="item in
cityList" :value="item.value" :key="item.value">{{ item.label }}</Option>
4.   </Select>
5. </template>
6.
7. <script setup>
8.   import { ref } from 'vue';
9.
10.  const cityList = ref([
11.    {
12.      value: 'New York',
13.      label: 'New York'
14.    },
15.    {
16.      value: 'London',
17.      label: 'London'
18.    },
19.    {
20.      value: 'Sydney',
21.      label: 'Sydney'
22.    },
23.    {
24.      value: 'Ottawa',
25.      label: 'Ottawa'
26.    },
27.    {
28.      value: 'Paris',
29.      label: 'Paris'
30.    },
31.    {
32.      value: 'Canberra',
33.      label: 'Canberra'
34.    }
35.  ]);
36.
37.  const model1 = ref('');
38. </script>
```

请选择 ^

New York

London

Sydney

Ottawa

Paris

Canberra

1. View UI Plus

1.5 表单组件

- DatePicker日期选择器
 - 日期选择器是一个选择或输入日期，支持年、月、日期等类型，支持选择范围。

```
1. <template>
2.   <Row>
3.     <Col span="12">
4.       <DatePicker type="date" placeholder="Select date" style="width: 200px"></DatePicker>
5.     </Col>
6.     <Col span="12">
7.       <DatePicker type="daterange" placement="bottom-end" placeholder="Select date" style="width: 200px"></DatePicker>
8.     </Col>
9.   </Row>
10. </template>

11. <script setup>
12.   import { Row, Col } from 'ant-design-vue';
13.   import { DatePicker } from 'ant-design-vue';
14. </script>
```



2. Vite

2.1 Vite简介

- Vite是一种新型前端构建工具，能够显著提升前端开发体验。
- 主要由两部分组成开发服务器和构建指令。
- 开发服务器基于原生 ES 模块提供丰富的内建功能，如快速模块热更新（HMR）。
- 构建指令使用预配置的 JavaScript 打包器（Rollup.js） 打包代码，输出用于生产环境的高度优化过的静态资源。
- Vite 还提供强大的扩展性，可通过插件 API 和 JavaScript API 进行扩展和项目优化，并提供完整的类型支持。
 - 极速的服务启动：使用原生ESM文件，无需打包。
 - 轻量快速的热重载：无论应用程序大小如何，都始终极快的模块热重载（HMR）。
 - 丰富的功能：对 TypeScript、JSX、CSS 等支持直接引用。
 - 优化的构建：可选“多页应用”或“库”模式的预配置Rollup.js构建。
 - 通用的插件：在开发和构建之间共享Rollup-superset插件接口。
 - 完全类型化的API：灵活的API和完整TypeScript类型。



2. Vite

2.1 Vite简介

- Vite 和传统的打包工具（如 webpack）在构建方式、编译方式、热更新方式、插件化方式和支持的框架等方面都有所不同。Vite 更加轻量、快速、灵活，适合于开发小型应用和组件库，而 webpack 更加适合大型应用的构建和优化。
- Vite 和传统的打包工具（如 webpack）有以下不同点。
 - 构建方式：Vite 采用基于浏览器原生 ES 模块的开发模式，在浏览器请求对应模块时，即时地编译和执行对应的代码，而 webpack 在构建时将所有模块打包成一个或多个bundle.js文件。
 - 编译方式：Vite 根据需要动态地编译模块，而 webpack 将所有模块都打包到一个文件中。
 - 热更新方式：Vite 支持热更新，可以在开发时实时更新修改后的代码，而 webpack 需要重新编译整个应用才能看到修改后的效果。
 - 插件化方式：Vite 支持插件化，可以通过插件扩展 Vite 的功能，而 webpack 则需要通过 loader 和 plugin 扩展其功能。
 - 支持的框架：Vite 支持多种前端框架，包括 Vue、React、Angular 等，而 webpack 则需要通过相应的 loader 和 plugin 来支持不同的框架。



2. Vite

2.1 Vite简介

- 项目构建是软件开发过程中的一个重要环节，项目构建是基于代码转换、资源优化、依赖管理、环境变量配置、自动化任务、模块化等，具体解释如下。
 - 代码转换：现代Web开发经常涉及到使用ES6+、TypeScript、CSS预处理器（如Sass、Less）、JSX等高级语言或语法。这些高级特性虽然提高开发效率和代码的可维护性，但浏览器并不直接支持。项目构建过程需要将这些高级代码转换为浏览器能够理解的格式（如JavaScript、CSS）。
 - 资源优化：在开发过程中，为调试和模块化开发，通常会使用多个JS、CSS文件、图片、字体等资源文件。但在生产环境下，过多的网络请求会导致页面加载速度变慢。项目构建过程可以对这些资源进行优化（合并文件、压缩代码、图片优化等），减少请求次数和文件大小，提高页面加载速度。
 - 依赖管理：引入的第三方库和框架依赖项的管理变得复杂且容易出错。Vite提供依赖管理功能，可以自动解析和加载项目中的依赖项，确保正确性和一致性。
 - 环境变量：不同的环境（如开发环境、测试环境、生产环境）需要不同的配置。Vite构建项目根据当前环境设置不同的环境变量，实现灵活的配置管理。
 - 自动化任务：Vite构建项目会执行自动化任务，如代码格式化、代码检查、单元测试等。提高开发效率，减少人为错误。
 - 模块化：现代Web开发强调模块化，即将代码分割成可复用的模块。Vite支持模块化开发，可以自动处理模块之间的依赖关系，确保代码的正确组织和加载。



2. Vite

2.2 Vite配置

- 在 Vite 项目中，构建配置涉及多个方面，vite.config.js是Vite的配置文件，其常见的配置项如下。
 - vite.config.js文件配置
 - base: 指定项目的公共基础路径。例如，如果设置为 “/my-app/”，则所有资源的URL都会基于此路径。
 - server:
 - port: 定义开发服务器监听的端口号。
 - host: 指定服务器运行的主机，例如 “localhost” 或特定的IP地址。
 - https: 启用或禁用HTTPS协议。
 - proxy: 配置代理规则，用于解决跨域请求问题。
 - build:
 - outDir: 明确构建输出的目录，默认是 “dist”。
 - assetsDir: 指定静态资源（如图片、字体）在输出目录中的子文件夹名称。
 - minify: 选择代码压缩的方式，“terser” 提供更精细的控制，“esbuild” 速度更快。
 - rollupOptions: 深入定制Rollup的构建选项，例如控制代码拆分、模块格式等。
 - plugins:
 - 例如，“vite-plugin-vue” 用于支持Vue框架，“vite-plugin-svg-icons” 用于优化SVG图标处理。



2. Vite

2.2 Vite配置

- 在 Vite 项目中，构建配置涉及多个方面，vite.config.js是Vite的配置文件，其常见的配置项如下。
 - 处理 CSS
 - 使用 sass : 安装sass相关依赖后，通过 “vite.config.js” 中的css对象进行配置，指定预处理器选项。
 - 引入全局 CSS : 可以在配置中指定全局样式文件的路径，确保在项目的每个组件中都能应用。
 - 处理静态资源
 - 图片: 可以设置不同大小图片的加载策略，如懒加载。
 - 字体: 配置字体文件的处理方式，包括字体格式的转换和优化。
 - 环境变量
 - 通过在项目根目录创建 “.env” 文件来设置不同环境的变量。例如，“.env.development” 用于开发环境，“.env.production” 用于生产环境。
 - 在代码中使用 “import.meta.env.VARIABLE_NAME” 来获取环境变量的值。
 - 自定义别名
 - 例如，将 “@/” 定义为 “src/” 的别名，方便在代码中更简洁地引用项目中的模块和文件。
 - 通过对这些构建配置的精细调整和优化，使Vite更好地适应项目的特定需求，提高开发效率和构建产物的质量。

2. Vite

2.2 Vite配置

● 以下为Vite配置的具体使用示例

```
1. import { defineConfig } from 'vite';
2. import vue from '@vitejs/plugin-vue';
3.
4. export default defineConfig({
5.   plugins: [vue()],
6.   root: './',
7.   base: '/my-app/',
8.   publicDir: 'public',
9.   resolve: {
10.    alias: {
11.      '@': '/path/to/src',
12.    },
13.    extensions: ['.js', '.ts', '.jsx', '.tsx', '.json', '.vue'],
14.  },
15.  server: {
16.    host: 'localhost',
17.    port: 3000,
18.    https: false,
19.    proxy: {
20.      '/api': {
21.        target: 'https://example.com',
22.        changeOrigin: true,
23.        rewrite: (path) => path.replace(/^\/api/, ''),
24.      },
25.    },
26.  },
27.  build: {
28.    outDir: 'dist',
29.    minify: 'terser',
30.    terserOptions: {
31.      compress: {
32.        drop_console: true,
33.        drop_debugger: true,
34.      },
35.    },
36.  },
37.});
```

2. Vite

2.3 配置优化

- Vite构建配置优化是提升应用性能、改善用户体验、提高开发效率、增强可维护性、支持现代Web技术、优化SEO、降低成本以及增强安全性的重要手段。通过合理配置和优化构建过程，可以使得项目更加高效、可靠和易于管理。以下是构建配置优化对应用的影响：
 - 提升加载速度：优化构建配置减少最终打包文件的体积，使用户在访问网站或应用时需要下载的数据更少，加快页面的加载速度，提升用户体验。
 - 改善性能：通过代码分割、按需加载等优化手段，确保用户只加载当前页面或功能所需的代码和资源，减少不必要的计算和内存占用，改善应用的运行性能。
 - 提高构建效率：构建效率影响到开发人员的迭代速度。优化Vite的构建配置可以缩短构建时间，使开发人员能够更快地看到代码变更的效果，从而提高开发效率。
 - 增强可维护性：合理构建配置使项目的结构更清晰，代码模块化，便于迭代和维护。通过自动化的构建和压缩过程，减少手动操作带来的错误和不一致性。
 - 支持现代Web技术：Vite作为现代前端构建工具，支持ES模块、HTTP/2、服务器推送等现代Web技术。通过优化构建配置，可以更好地利用这些技术带来的优势，提升应用的性能和用户体验。

2. Vite

2.3 配置优化

- Vite构建配置优化是提升应用性能、改善用户体验、提高开发效率、增强可维护性、支持现代Web技术、优化SEO、降低成本以及增强安全性的重要手段。通过合理配置和优化构建过程，可以使得项目更加高效、可靠和易于管理。以下是构建配置优化对应用的影响：
 - 提升SEO：较快的加载速度和优化的代码结构对搜索引擎优化（SEO）也有积极的影响。搜索引擎更倾向于排名那些加载速度快、内容组织良好的网站。
 - 降低带宽和存储成本：对于需要部署到云服务器或CDN的应用，较小的包体积意味着更低的带宽消耗和存储成本。
 - 增强安全性：通过移除生产环境中的调试信息（如console.log语句）和不必要的代码，可以减少潜在的安全风险。合理的构建配置可以帮助实现资源的HTTPS传输等安全措施。

2. Vite

2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 使用gzip压缩
 - gzip压缩可以显著减少文件大小，加快文件传输速度。在Vite中，可以通过安装“vite-plugin-compression”插件来实现gzip压缩

```
1. import viteCompression from 'vite-plugin-compression';
2.
3. export default defineConfig({
4.   plugins: [
5.     viteCompression({
6.       verbose: true, //表示在构建过程中显示详细的压缩信息。
7.       disable: false, //表示启用压缩功能。
8.       deleteOriginFile: false, //表示不删除原始未压缩的文件。
9.       threshold: 5120, // 表示只有大于或等于这个大小的文件才会被压缩。
10.      algorithm: 'gzip', //表示使用 gzip 算法进行压缩。
11.      ext: '.gz' //表示压缩后的文件扩展名为 .gz。
12.    })
13.  ]
14.});
```

2. Vite

2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 静态资源打包处理
 - 优化静态资源的打包处理，按分类存放相应的文件，可以提高缓存效率。在Vite中，可以通过配置 “build.rollupOptions.output” 来实现：

```
1. build: {
2.   rollupOptions: {
3.     output: {
4.       //控制动态导入的 chunk 文件的命名方式。
5.       chunkFileNames: 'static/js/[name]-[hash].js',
6.       //控制入口文件的命名方式。
7.       entryFileNames: 'static/js/[name]-[hash].js',
8.       // 控制静态资源（如图片、字体等）的命名方式。
9.       assetFileNames: 'static/[ext]/[name]-[hash].[ext]',
10.      // 定义一个函数用于手动指定哪些模块应该被打包到同一个 chunk 中。
11.      manualChunks(id) {
12.        if (id.includes('node_modules')) {
13.          return id.toString().split('node_modules/')[1].split('/')[0].toString();
14.        }
15.      }
16.    }
17.  }
18.}
```

2. Vite

2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 代码分割与大文件拆分
 - 代码分割可以将代码拆分成多个包，按需加载，减少初始加载时间。在Vite中，可以通过配置 “build.chunkSizeWarningLimit” 和 “manualChunks” 来实现：

```
1. build: {
2.   chunkSizeWarningLimit: 1500, // 控制 chunk 大小警告的阈值。
3.   //定义一个对象来手动指定哪些模块应该被打包到同一个 chunk 中
4.   manualChunks: {
5.     vue: ['vue', 'vue-router'],
6.     lodash: ['lodash'],
7.     // 可以根据实际需要合并其他库或组件
8.   }
9. }
```

2. Vite

2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 组件按需导入
 - 使用“unplugin-vue-components”插件可以实现Vue组件的按需导入，减少最终打包体积。安装并配置该插件后，Vite会在构建时自动分析并导入用到的组件。

```
1. // vite.config.js
2. import { defineConfig } from 'vite';
3. import Components from 'unplugin-vue-components/vite';
4. import { AutoImport } from 'unplugin-auto-import/vite';
5. import { createResolver } from 'unplugin-vue-components/resolvers';
6. // 定义一个解析器来处理 iView 的组件
7. const iViewResolver = createResolver({
8.   // iView 的组件路径前缀
9.   prefix: 'iView',
10.  // 匹配组件的正则表达式
11.  resolveComponentPath: (name) => `iView/es/${name}/index`,
12. });
13. export default defineConfig({
14.  plugins: [
15.    // 自动导入 Vue 组件
16.    Components({
17.      resolvers: [
18.        iViewResolver,
19.      ],
20.      dts: true, // 开启生成 .d.ts 声明文件的支持
21.    }),
22.    // 自动导入 Vue 的 Composition API
23.    AutoImport({
24.      imports: ['vue', 'vue-router', 'vuex'], // 可以根据需要添加其他库
25.      resolvers: [],
26.    }),
27.    // 其他插件...
28.  ],
29.  // 其他配置...
30. });
```

2. Vite

2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 清除console和debugger
 - 在生产环境中，移除console.log和debugger语句可以减小文件体积并避免潜在的安全风险。可以通过配置Terser插件来实现：

```
1. build: {
2.   terserOptions: {
3.     compress: {
4.       // 设置为 true 表示在压缩过程中移除所有的 console 调用。
5.       drop_console: true,
6.       // 设置为 true 表示在压缩过程中移除所有的 debugger 语句。
7.       drop_debugger: true
8.     }
9.   }
10. }
```

2. Vite

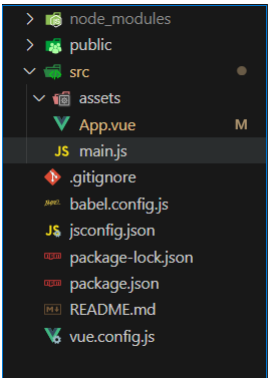
2.3 配置优化

- Vite 构建配置优化是提升前端项目构建效率和性能的关键步骤。以下是一些常见的Vite构建配置优化方法：
- 使用CDN加速
 - 对于第三方库或大型资源，可以考虑使用CDN来加速资源加载。Vite中可以通过安装vite-plugin-cdn-import插件来实现：

```
1. import { importToCDN } from 'vite-plugin-cdn-import';
2.
3. export default defineConfig({
4.   plugins: [
5.     importToCDN({
6.       modules: [
7.         // 需要CDN加速的模块 假设为lodash
8.         // name: 库的名称。
9.         // var: 在全局变量中注册的名称。
10.        // path: CDN 链接地址
11.        {
12.          name: 'lodash',
13.          var: '_',
14.          path: 'https://cdn.jsdelivr.net/npm/lodash@4.17.21/lodash.min.js'
15.        }
16.      ]
17.    })
18.  ]
19.});
```

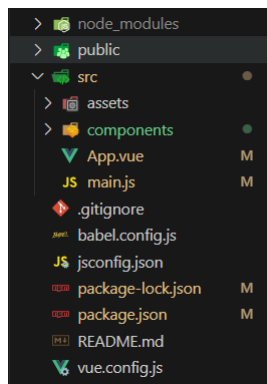
3. 使用View UI Plus实现管理系统

- 步骤1: 创建名为“manage”的项目
- 步骤2: 打开“manage”项目，删除“src\assets”、“src\components”目录下的示例文件。
- 步骤3: 打开终端输入“npm install view-ui-plus --save”安装View UI Plus。



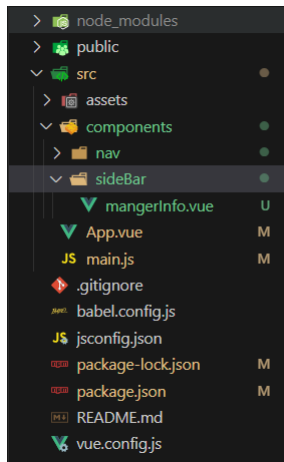
3. 使用View UI Plus实现管理系统

- 步骤4: 在“src\components”目录下创建管理系统导航栏文件夹“src\components\nav”，其中包含了两个文件，分别是“src\components\nav\personnalInfo”和“src\components\nav\addMangerInfo”。



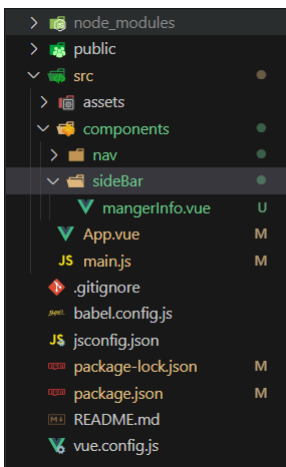
3. 使用View UI Plus实现管理系统

- 步骤5: 在“src\components”目录下创建管理系统侧边栏文件夹“src\components\sideBar”，目前只用到一个侧边栏所以只有一个文件“src\components\sideBar\mangerInfo”。



3. 使用View UI Plus实现管理系统

- 步骤6：编辑 “App.vue”文件。



信创智能医疗系统研发课程体系
河南中医药大学信息技术学院（智能医疗行业学院）



河南中医药大学信息技术学院（智能医疗行业学院）智能医疗教研室
河南中医药大学医疗健康信息工程技术研究所