数据库开发前端指南

by 2151938 杨昕迪

目录

1. 引言部分

本文档是对前两周所有有关前端部分的修改与汇总，方便大家查阅。

前端三件套（HTML，CSS，JavaScript)）我个人认为对于我们这次项目来说，最重要的是 html，这个不会就写不了一点。其次 css 需要做总的样式表（比如说某些元素有没有边框，边框有多宽等等）。js 的话反而不那么重要，vue 提供的对 dom 元素的修改方式与普通的 js 不同，普通的 js 是为元素起名，然后用 getElementByld 等等函数去找到这些元素，然后修改；vue,是绑定元素的 id 然后为其定义属性和方法，用定义的方法去修改定义的属性，然后在 html 上呈现，因此 js 仅需要简单了解(其实和 c++语法很接近，主要是变量那里不太一样.并且涉及到一些表单之类的验证功能)。

我更推荐大三是网络的同学来做前端，网络大三做项目需要 vue(前端+springboot(后端，属于 Java，有兴趣可以暑假了解一下)，现在可以提前学习 vue 为以后做准备。

2. Vue 部分

2.1 版本选择

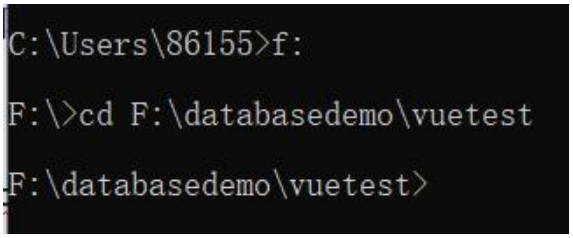
经过资料的查找，我们决定使用 **VSCode+Vue3+Element Plus**（Element Ui 的 Plus 版，Vue2 适配 Element Ui，Vue3 适配 Element Plus），麻烦做前端的同学安装一下 VSCode，VSCode 针对于编写现代 Web 和云应用的跨平台源代码，支持一系列其他语言，如 C++，Python 等等，并提供丰富且功能十分强大的插件辅助编程。具体安装文档请参考之前发在群里的安装文档。

Element Ui 的网络资料要多于 View Ui，且与 Vue 完美适配，因此不再使用 View Ui；VSCode+Vue3+Element Ui（Plus）的网络资料也多于 VS2019，此外，抛开本次项目为长远考虑，我曾咨询过高年级学长，学长都很推荐使用 VSCode，所以再次麻烦做前端的同学根据文档安装（没安装的话麻烦看一下我第一周发在群里的文档，因不是我的版权，所以不再上传 QQ 群）。

2.2 创建新的 Vue 工程项目

下面演示 Vue 项目的创建：

1. 在 cmd 窗口执行 npm install webpack –g 与 npm install vue-cli -g 两句命令，我个人是在基地址执行的，也就是在左下角直接打开 cmd，并没有更改根地址执行（由于我已经执行过，所以具体的显示和大家不太一样）。如果大家这里执行有问题的话，也可以在步骤 2 中创建的文件夹这一地址下执行这两句命令；
2. 进入存放项目的文件夹，我是 vuetest；



3. 进入后输入命令 vue create 项目名，此处项目名是 vue01；

```
F:\databasedemo\vuetest>vue create vue01
```

4. 用下箭头↓移到第三个选项 Manually select features 并按回车；

```
Vue CLI v5.0.8
? Please pick a preset:
  Default ([Vue 3] babel, eslint)
  Default ([Vue 2] babel, eslint)
> Manually select features
```

5. 用上下箭头移到以下三个选项，按空格键添加，添加后按回车确定；

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to toggle all, <i> to invert selection, and
<enter> to proceed)
 (*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 (*) Router
 (*) Vuex
 ( ) CSS Pre-processors
 ( ) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing
```

6. 选择 3.x 并按回车；

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 3.x
  2.x
```

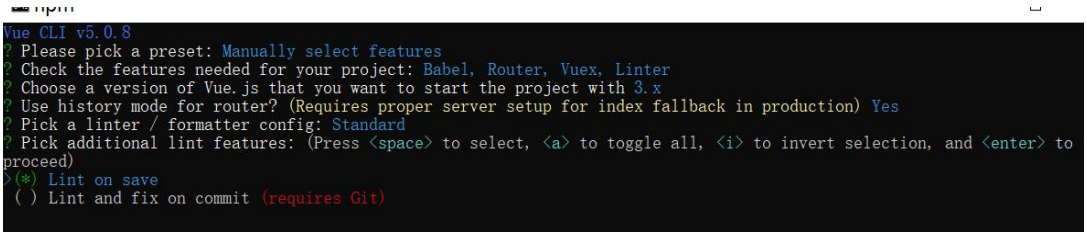
7. 输入 y 并按回车；

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n)
```

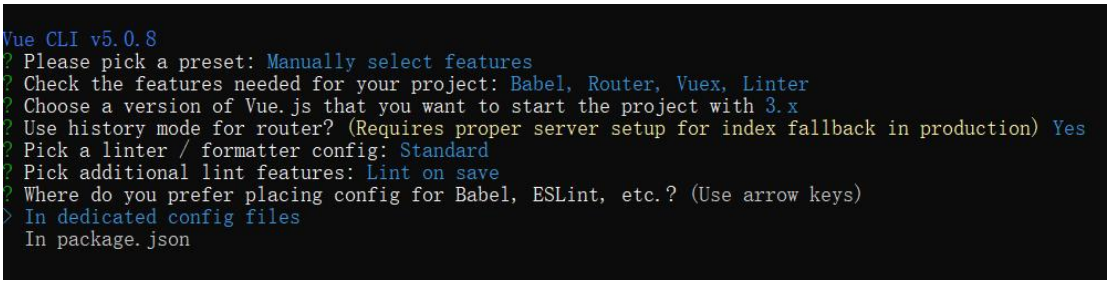
8. 用下箭头移到第三个选项 ESLint + Standard config 并按回车；

```
Vue CLI v5.0.8
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, Router, Vuex, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a linter / formatter config:
  ESLint with error prevention only
  ESLint + Airbnb config
> ESLint + Standard config
  ESLint + Prettier
```

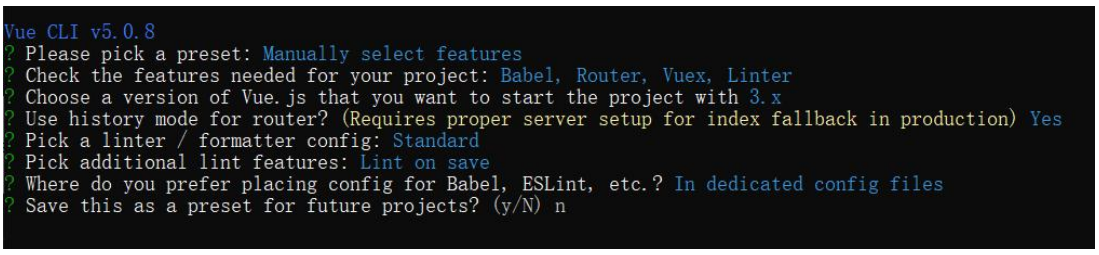
9. 我们用第一个选项 Lint on save，直接按回车；



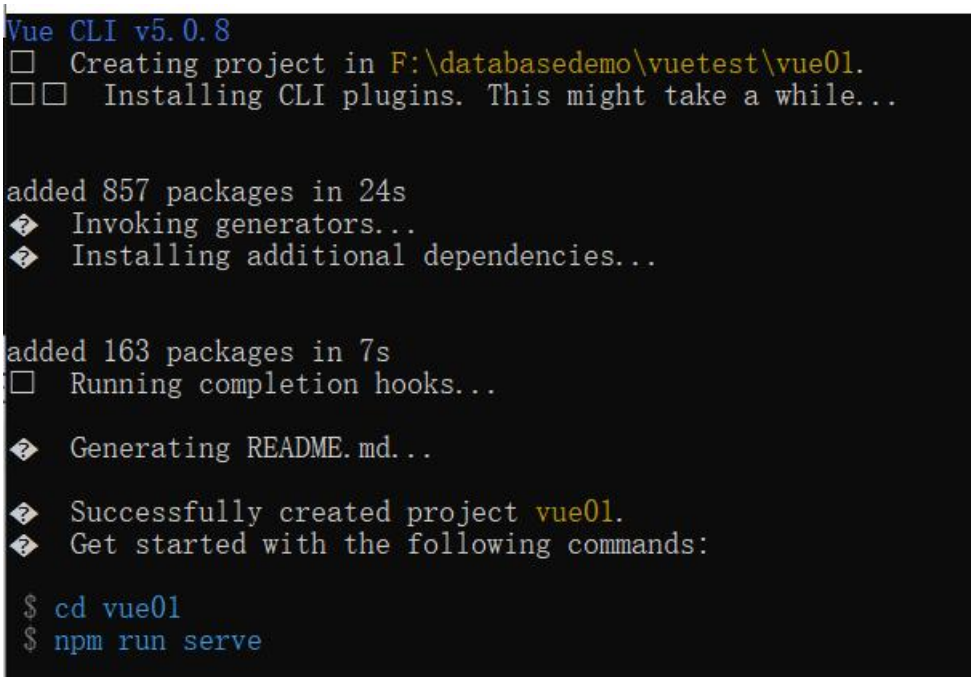
10. 我们选择第一个选项 In dedicated config files，直接按回车；



11. 输入 n 并按回车；

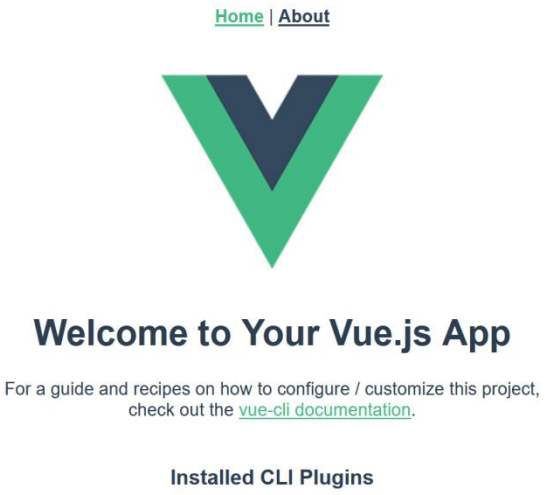


成功后是这样的：



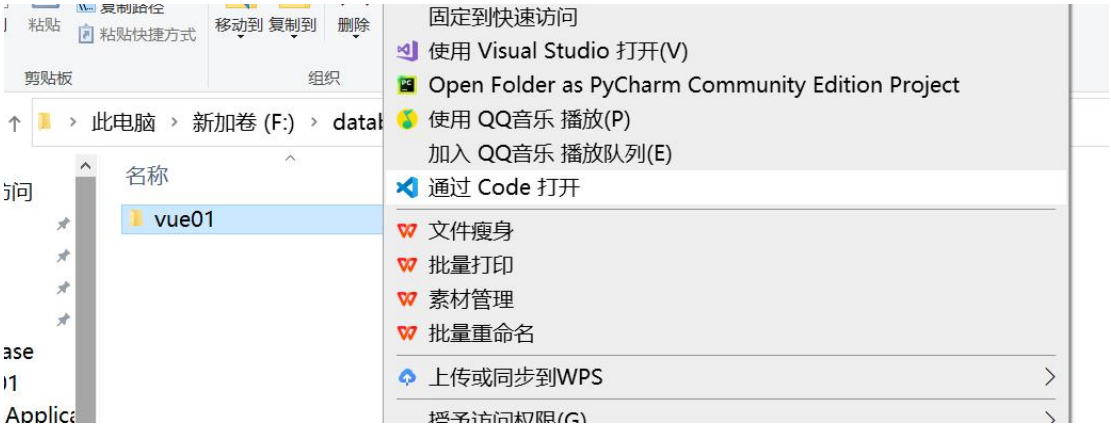
输入 cd vue01（进入所创建文件夹）以及 npm run serve 两条命令，出现以下提示，打开后网页成功加载证明创建完成（调整 Network 的 unavailable 请参考刘天琦同学之前发的笔记，因不影响本地调试，此处不再给出调整方法）。

```
F:\databasedemo\vuetest>cd vue01

F:\databasedemo\vuetest\vue01>npm run serve

> vue01@0.1.0 serve


 DONE  Compiled successfully in 6551ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: unavailable

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

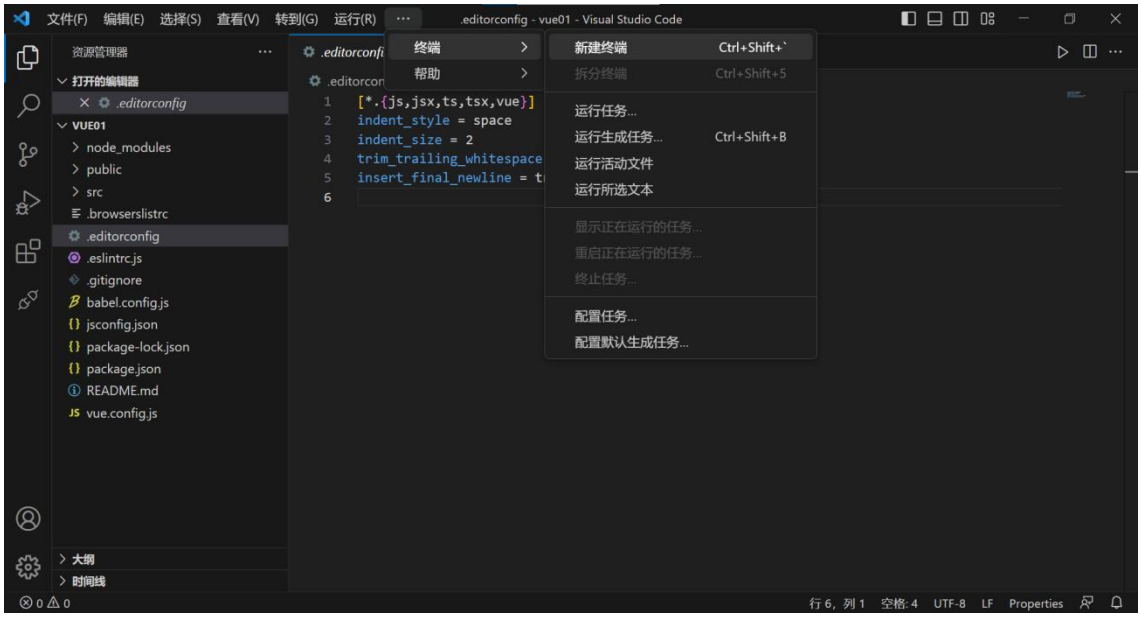


2.3 用 VSCode 打开项目

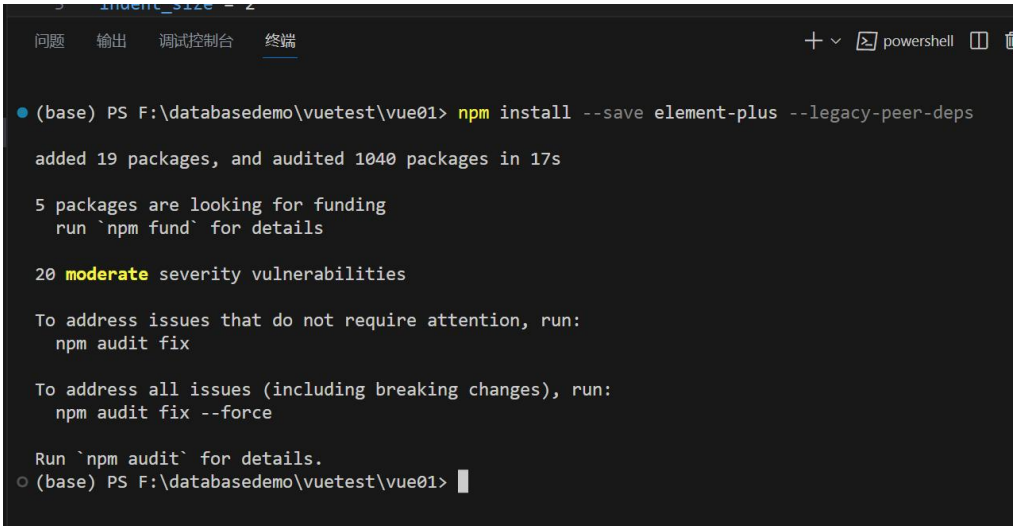
1. 用 VSCode 打开文件夹；



2. 新建一个终端（就相当于 cmd，和 cmd 没有任何区别）；



3. 在终端输入指令 npm install --save element-plus --legacy-peer-deps 并按回车，出现以下部分表示安装成功



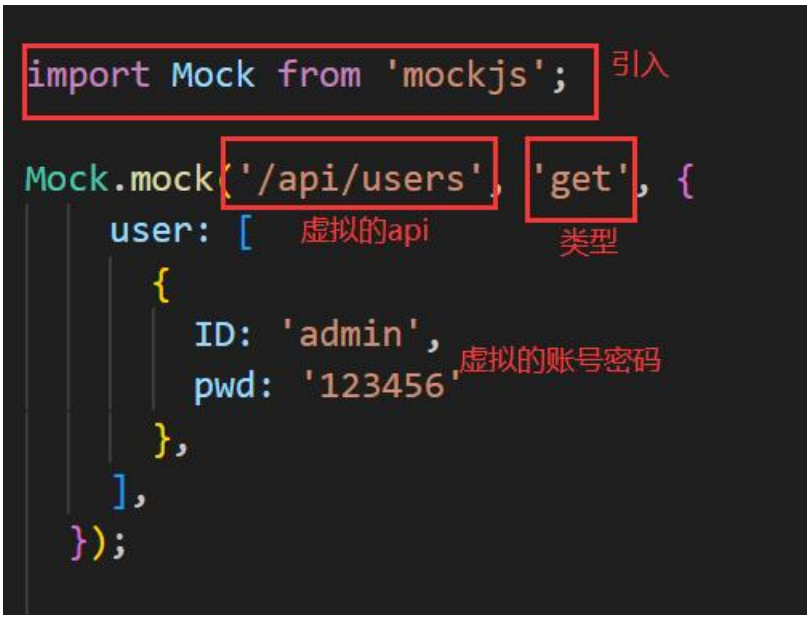
2.4 用 mockjs 模拟假数据

Mock.js 是一个前端开发中常用的模拟数据生成工具，它可以帮助前端开发人员在开发过程中模拟接口返回的数据。在前后端分离的开发中，前端开发人员通常需要等待后端接口的完成才能进行开发和测试。使用 Mock.js 可以在后端接口尚未实现的情况下，模拟生成符合接口预期数据结构的数据，以便前端开发人员进行界面的开发和调试。

1. 安装

执行指令 npm i mockjs -D

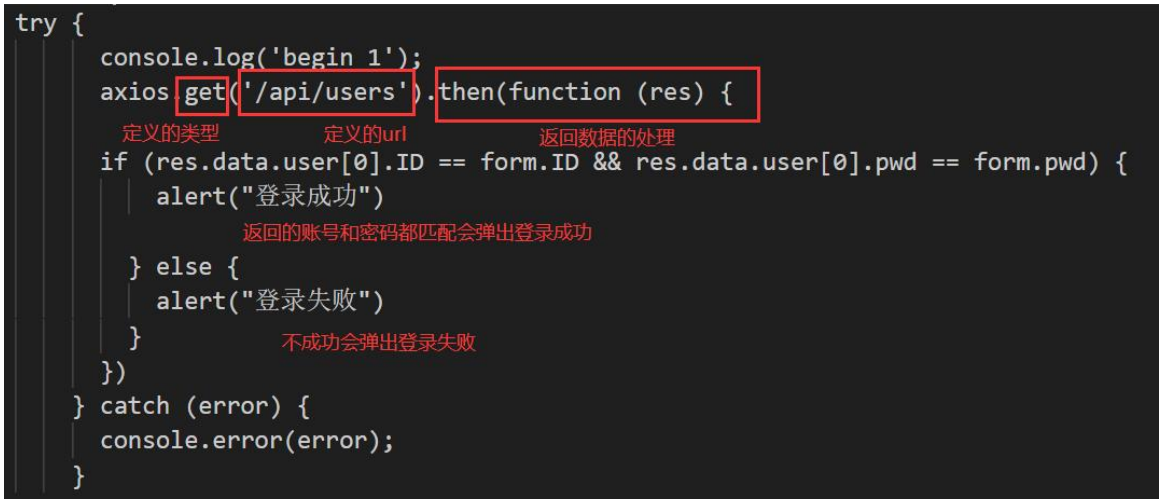
2. 引入及定义数据



3. 开启 mock

注意：2.1 与 2.2 都在 main.js 中添加即可

4. 前端使用



mockjs 只作为与后端分离时使用，我们目前所实现的只有在本机（即前端和后端都在同一台笔记本上）的连接，因此现阶段可以先使用 mock 来测试。在后端部署到云服务器后前端可以与云服务器连接，就不需要 mockjs 了。

2.5 router 跳转不同界面

router 指的是前端路由。
ChatGPT 生成：
1. 前端路由可以根据用户的交互或应用程序的状态，通过改变 URL 或路由路径来导航到不同的页面或视图。它允许用户在应用程序中进行页面间的导航，而无需刷新整个页面。通过前端路由，用户可以直接访问不同的视图，提供了更流畅的用户体验。

2. 页面切换和渲染：前端路由能够捕获 URL 的变化，并将其与预定义的路由规则进行匹配，以确定要渲染的页面或组件。通过前端路由，应用程序可以根据 URL 的变化动态加载并渲染相应的页面，使用户能够在应用程序中浏览不同的视图。

3. 状态管理：前端路由可以帮助管理应用程序的状态。例如，可以使用路由参数来传递数据、设置页面的状态或标识当前活动的页面。通过路由状态的管理，应用程序可以根据需要展示不同的内容或执行不同的操作。

4. 嵌套和组织视图：前端路由支持嵌套路由，可以将应用程序的页面和视图层次结构进行组织和管理。这样可以更好地划分应用程序的功能模块，使代码更加模块化和可维护。

5. 历史记录和导航管理：前端路由通常会维护一个历史记录，记录用户的浏览历史。这样用户可以通过前进和后退按钮在历史记录中导航，返回之前访问过的页面。通过前端路由，应用程序可以提供类似于浏览器的导航体验，增强用户的操作控制性。

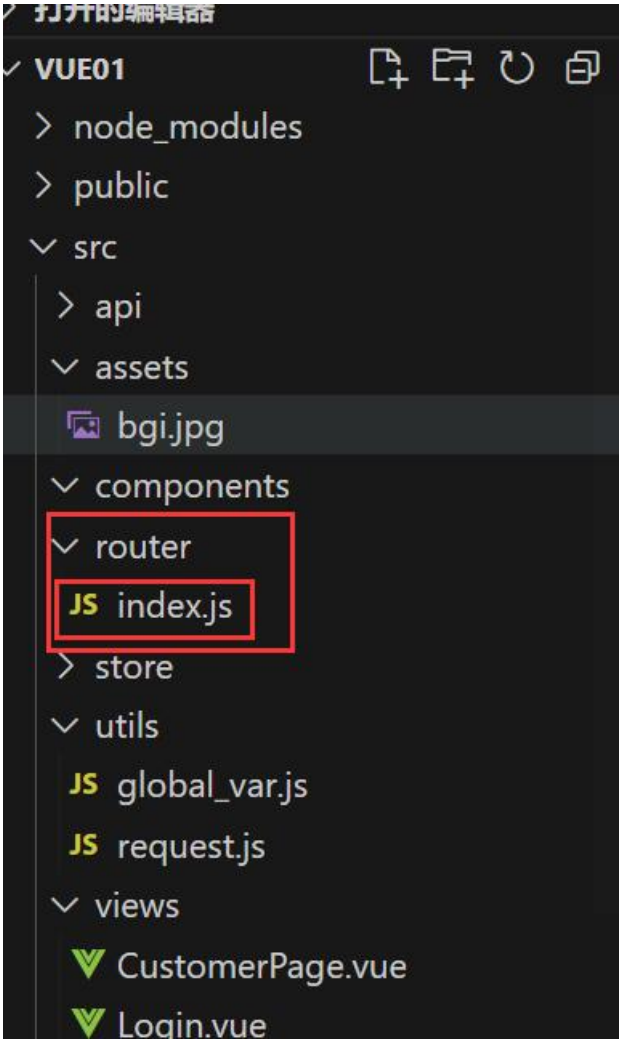
总之，在我们的项目开发中的作用主要是跳转到其他界面。本文档仅是基本的使用方法，如果需要更高级的功能请参考官网。

使用方法：
1. 在 main.js 中引入并挂载，如下图所示：

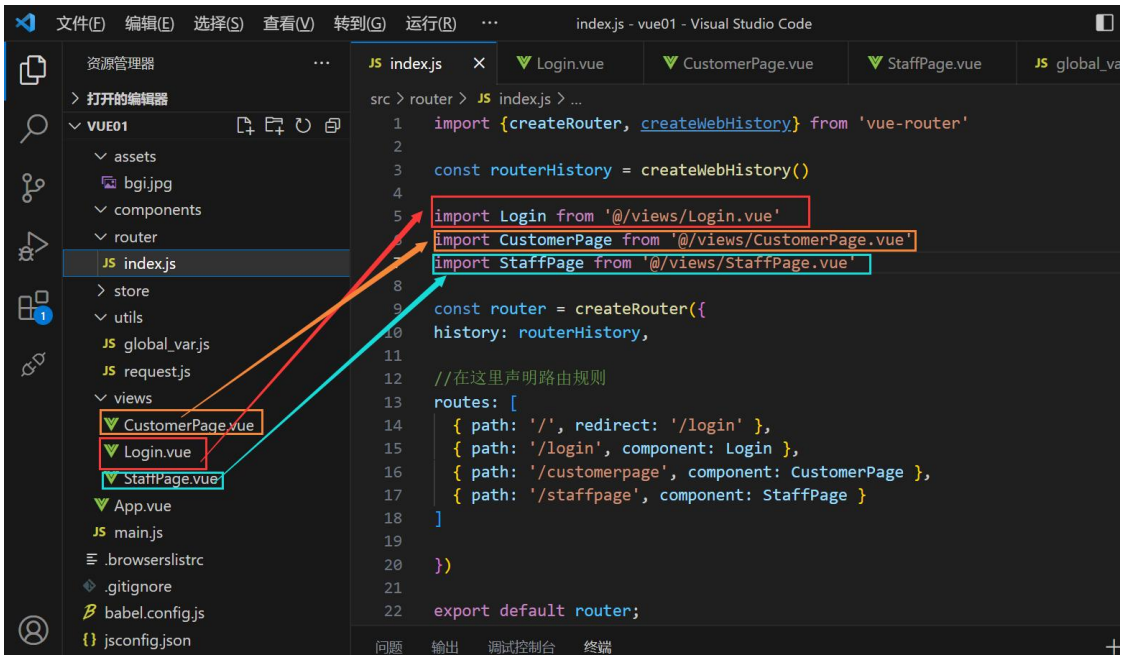
```
main.js > ...
import { createApp } from 'vue';
import App from './App.vue';
import router from './router';
import store from './store';
import ElementPlus from 'element-plus';
import 'element-plus/dist/index.css';


import Mock from 'mockjs';
```

```
createApp(App)
  .use(store)
  .use(ElementPlus)
  .use(router)
  .mount('#app');
```

2. 找到 src 下的 router 下的 index.js，如下图所示：



3. 根据语法，将文件映射到组件，如下图所示：



以 import Login from '@/views/Login.vue' 为例，Login 是自己起的组件名，'@/views/Login.vue' 是文件的真实地址。

4. 根据语法声明路由规则，如下图所示：

```
//在这里声明路由规则
routes: [
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  { path: '/customerpage', component: CustomerPage },
  { path: '/staffpage', component: StaffPage }
]
```

首句指定了刚打开我们的前端时到达的地址，后面的语句指定了组件的使用地址，以 path: '/customerpage', component: CustomerPage 为例，在使用时应该写 router.push({ path: "/customerpage" });也就是说要写 path 这一字符串。

```
router.push({ path: "/customerpage" });
```

这里的规则实际上是将组件映射到 path 上，这里四句话第二句是将组件 Login 映射到 path 的’/login’，第一句是使用了’/login’这一 path 作为初始页面的地址，第三第四句话同第二句话。

5. App.vue 的 template 模板中必须有<router-view></router-view>，否则无法跳转，如下图所示：

```
src > App.vue > template
1   <template>
2     <div>
3       <router-view></router-view>
4     </div>
5   </template>
```

6. 这样实操时通过 router.push 即可跳转。

2.6 实例

此处提供通过 mockjs 制造假数据，用户在前端登录跳转到不同界面的实例解析，代码已上传压缩包。

下面我将展示如何实现在登录界面实现输入账号密码登录成功后跳转到不同界面。由于前后端的进度不同，我将使用 mock 生成数据方便调试。

mock 数据如下图所示：

```
Mock.mock('/api/login', 'get', () => {
  // 生成随机的登录结果，success 或 fail
  const loginResult = Mock.mock('@boolean');

  // 生成随机的身份值，1 或 2 或 3
  const identity = Mock.mock('@integer(1, 3)');

  // 返回模拟的数据
  return {
    success: loginResult ? 'success' : 'fail',
    identity: identity
  };
});
```

每次给前端返回的 response 有两个数据，第一个是 loginResult，会返回 success 或 fail 两个字符串表示登录成功与否（此处需后端判断后返回相应字符串），以及一个身份信息 1 或 2 或 3，1 代表销售部，2 代表技术部，3 代表买家。

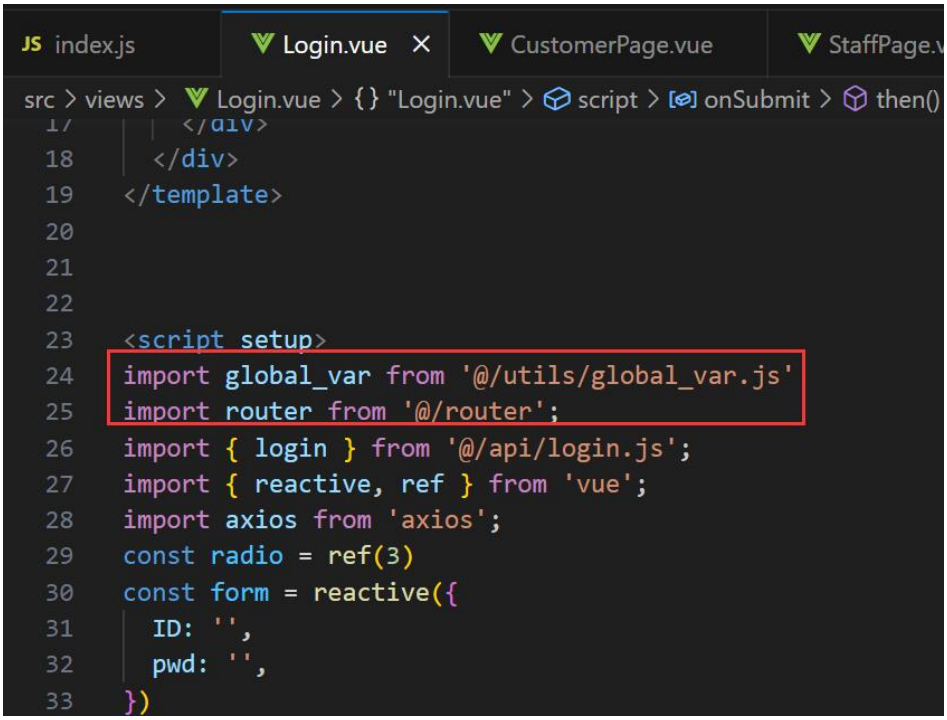
我在 src 的 utils 的 global_var.js 中定义了全局变量，来记录当前用户的身份信息，如下图所示：

```
src > utils > JS global_var.js > default
1   /*全局变量*/
2   export default{
3       identity:-1,//身份    -1为默认，1为销售部，2为技术部，3为买家
4   }
```

路由的设置与上文一样，不再赘述。

在打开前端编译后，打开的地址会直接到 login，因为前文已经将初始地址设置到 login 界面。

在 login 的.vue 文件的 script 部分需要引入以下：



这样才可以顺利使用全局变量和路由。全局变量的使用方法以图中为例是 global_var.identity。

在登录界面输入账号密码后，会向 mock 发 get 请求，mock 随机返回结果，如下图所示：

```
axios.get('/api/login', { username: form.ID, password: form.pwd })
.then(response => {
  const data = response.data;

  // 处理返回的模拟数据
  if (data.success === 'success') {
  // 登录成功
    alert("登录成功");
    console.log('登录成功');
    console.log('身份：', data.identity);
    global_var.identity=data.identity;
    console.log(global_var.identity);
    if(global_var.identity==1){
      router.push({ path: "/customerpage" });
    }
    else if(global_var.identity==2){
      router.push({ path: "/staffpage" });
    }
    else{
      router.push({ path: "/staffpage" });
    }
  } else {
    // 登录失败
    alert("登录失败");
    console.log('登录失败');
  }
```

给全局变量赋值，到时候会用它做权限区分

跳转不同地址

因为 mock 是随机返回的，所以实际测试需要多尝试几次来返回想要的数据。这样就可以跳转到相应的界面。

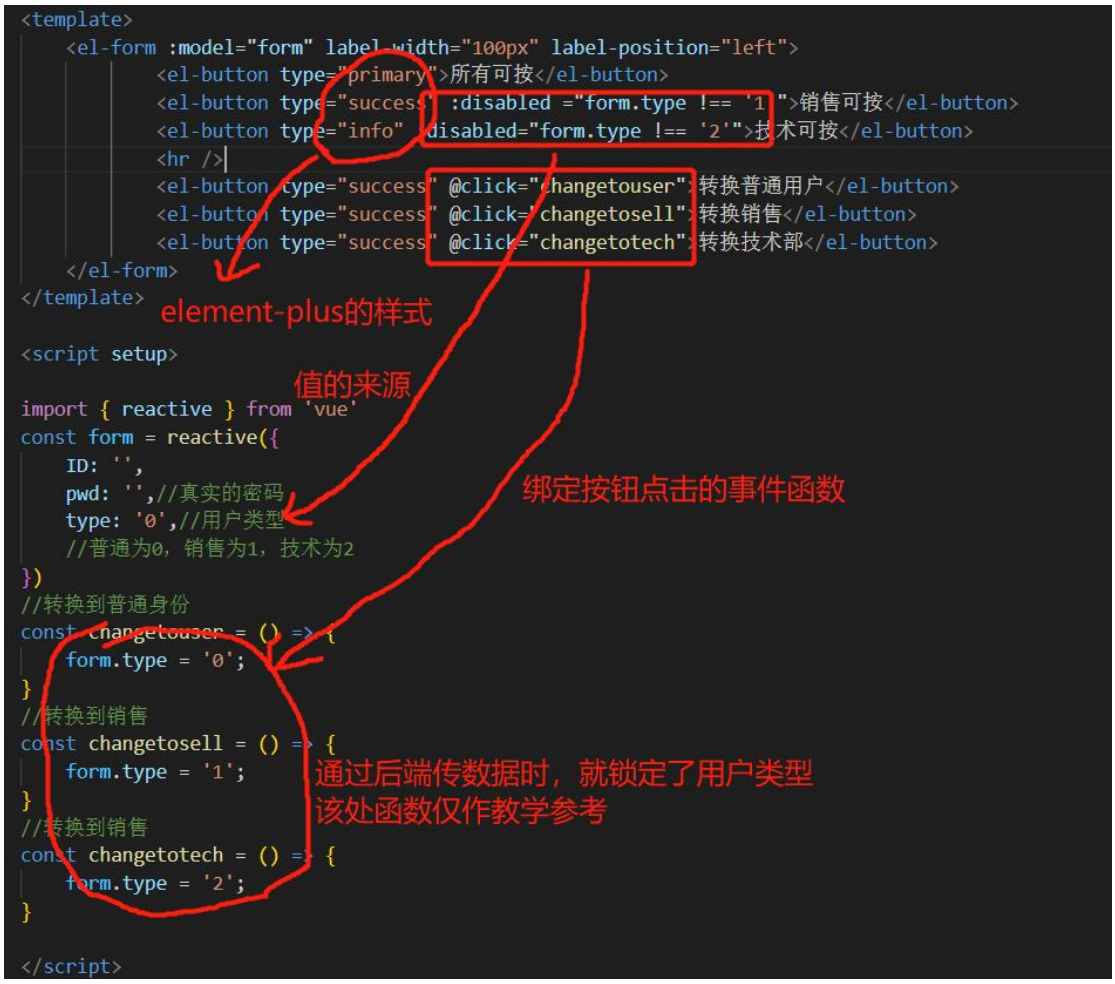
如图所示，身份值为 2，是技术部员工，只可以按所有可按和技术可按。

权限的区分由全局变量的身份值来确定，具体的区分方法由罗国蔚同学提供文档。

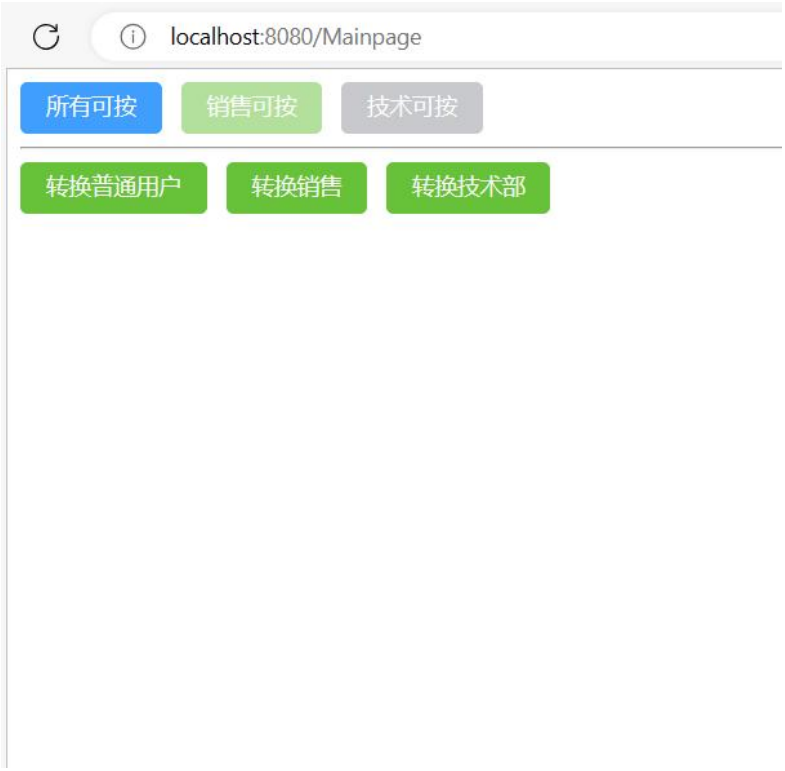
3. 附录

3.1 权限区分按钮简易教学--by 罗国蔚

代码详情见 Mainpage.vue



效果：（点击转换按钮后能改变身份从而能点击不同的按钮）


localhost:8080/Mainpage
所有可按
销售可按
技术可按
转换普通用户
转换销售
转换技术部