环境

```
node:
16.17.1

npm:
8.15.0
```

Ant Design of React官网：https://ant.design/docs/react/introduce-cn

一、创建项目

```
npm init vite

Project name: lege-management
Select a framework: react
Select a variant: react-ts
```

打开package.json，参考以下各模块版本：

```
"dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-redux": "^7.2.8",
    "react-router-dom": "^6.3.0",
    "redux": "^4.1.2"
},
```

下载后进入到项目文件夹，安装package.json中的包：

```
npm i
```

尝试更改package.json启动命令：

```
"scripts": {
    "dev": "vite --host --port 3002",
    "build": "tsc && vite build",
    "preview": "vite preview"
}
```

二、项目目录初始化

删除掉官方自带而对我们暂时帮助不大的文件。

删除src下除了main.tsxs和App.tsx的其他文件；

主文件/src/main.tsx修改成：

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

App.tsx文件修改成：

```
import { useState } from 'react'
function App() {
  const [count, setCount] = useState(0)
  return (
    <div className="App">
      顶级组件
    </div>
  )
}
export default App
```

三、样式初始化

【注：乐哥认为reset-css比Normalize.css更直接，干净利落去除默认样式，更适合在企业里的场景，所以用reset-css，而不用Normalize.css】

路径下执行以下命令，安装reset-css：

```
npm i reset-css
```

在src/main.tsx中引入reset-css：

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import "reset-css"
import App from './App'


ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

四、scss的安装和初步使用

安装sass：

```
# 安装sass   vite中很方便，loader这些都不用自己配置，只需要安装好即可使用
npm i --save-dev sass
```

src下新建assets/styles/global.scss：

```scss
$color:#eee;
body{
    // 禁止选中文字
    user-select:none;
    background-color: $color;
}
img{
    // 禁止拖动图片
    -webkit-user-drag:none;
}
```

main.tsx中引入全局样式

```tsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import "reset-css"
import "./assets/styles/global.scss"
import App from './App'


ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

五、配置项目路径别名

5.1、路径别名的配置

目前ts对@指向src目录的提示是不支持的，vite默认也是不支持的。

所以需要手动配置@符号的指向

在vite.config.ts中添加配置：

```ts
import path from "path"

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  resolve:{
    alias:{
      "@":path.resolve(__dirname,'./src')
    }
  }
})
```

这时候引入的会path模块报红，但其实我们已经有node，所以就已经有path模块，只是缺少ts的一些声明配置。

所以需要安装关于node这个库的ts声明配置

```
npm i -D @types/node
```

安装成功就没有报红了,如果import后面的path报红，就把引入换成 `import * as path from 'path';`

5.2、配置路径别名的提示

虽然现在路径别名已经有了，但是在文件中输入@是没有提示路径的

需要我们在tsconfig.json中：添加两项配置

```
"compilerOptions": {
    ...
    "baseUrl": "./",
    "paths": {
      "@/*": [
        "src/*"
      ]
    }
},
```

配置好之后敲@后就有路径资源提示了

六、scss模块化

6.1、scss的进一步使用

src下新建components文件夹

components文件夹下新建Comp1文件夹，新建index.tsx和comp1.scss

src/components/Comp1/index.tsx中：

```
import "./comp1.scss"

function Comp() {
  return (
    <div className="box">
        <p>这是Comp1里面的组件</p>
    </div>
  )
}
export default Comp
```

src/components/Comp1/comp1.scss中：

```
.box{
    color:red
}
```

components文件夹下新建Comp2文件夹，新建index.tsx:

```
// ！！！注意，在Comp2组件中不引入上面的comp1.scss样式
function Comp() {
  return (
    <div className="box">
        <p>这是Comp2里面的组件</p>
    </div>
  )
}
export default Comp
```

在App.tsx中使用这两个组件：

```
import { useState } from 'react'
import Comp1 from "./components/Comp1"
import Comp2 from "./components/Comp2"

function App() {
  const [count, setCount] = useState(0)

  return (
    <div className="App">
      <Comp1></Comp1>
      <Comp2></Comp2>
    </div>
  )
}

export default App
```

此时会发现，Comp2/index.tsx中没有引入comp1.scss竟然也有box的样式！！！

说明Comp1的引入comp1.scss就是全局引入，这不是我们要的现象，所以这时候我们就需要用到模块化css

6.2、scss的模块化

src/components/Comp1/comp1.scss 改名为 comp1.module.scss

在comp1.tss中：

```
// 修改引入的语句
import styles from "./comp1.module.scss";

function Comp() {
  return (
      // 修改类名编程模块化写法
    <div className={styles.box}>
        <p>这是Comp1里面的组件</p>
    </div>
  )
}
export default Comp
```

这样就不会影响其他组件了。

！！注意：styles.box 的 box 在 .module.scss 文件中只能是类名，标签名字不起作用

七、Antd Design初步引入

安装Antd Design

```
// 使用 npm 安装
npm install antd --save

// 使用 yarn 安装
yarn add antd
```

安装图标所需要的模块

```
// 使用 npm 安装
npm install --save @ant-design/icons

// 使用 yarn 安装
yarn add @ant-design/icons
```

App组件中引入即可使用：

```
import { useState } from 'react'

import { Button } from 'antd';
import {FastBackwardOutlined} from "@ant-design/icons"
import 'antd/dist/antd.css'; // or 'antd/dist/antd.less'

function App() {
  const [count, setCount] = useState(0)

  return (
    <div className="App">
      顶级组件
      <Button  type="primary">按钮文字</Button>
      <FastBackwardOutlined style={{ fontSize: '40px', color: '#08c' }}/>
    </div>
  )
}

export default App
```

ps: 如果你的vscode不兼容jsx语法提示：请打开vscode配置文件，补充：

```
"emmet.triggerExpansionOnTab": true,
    "emmet.includeLanguages": {
        "javascript": "javascriptreact"
}
```

八、配置Antd Design样式自动按需引入

antd的4.x版本以上已经支持组件按需引入，我们只需要解决样式上的自动按需引入即可

安装插件vite-plugin-style-import

```
npm install vite-plugin-style-import@1.4.1 -D
```

在vite.config.ts中进行配置：

```
import styleImport,{AntdResolve} from 'vite-plugin-style-import';

export default defineConfig({
  plugins: [
    react(),
    styleImport({
      resolves: [
        AntdResolve()
      ],
    }),
  ],
  ...
})
```

在去掉APP.vue中的`import 'antd/dist/antd.css'; // or 'antd/dist/antd.less'`这一行样式引入

启动项目，发现报错，缺少less，进行安装

```
npm i less@2.7.1 -D
```

九、React路由——第一种配置方案（旧项目中的写法）

9.1、初步展示

我们在这里模拟vue中的home和about两个组件展示

【1、准备界面】首先src下创建views文件夹，views文件夹下创建Home.tsx和About.tsx，大致代码如下：

```
function View() {
    return (
        <div className="home">
            <p>Home</p>
        </div>
    )
}
export default View
```

【2、配置对应关系】/src下新建router文件夹，再进去新建index.tsx

```
import App from "../App"
import Home from "../views/Home"
import About from "../views/About"
import {BrowserRouter,Routes,Route} from "react-router-dom"
// 两种路由模式的组件： BrowserRouter ( History模式 ) ， HashRouter( Hash模式 )

// const baseRouter = () => {
//     return ()
// }
```

```
// 以上写法可以简写为：
const baseRouter = () => (
    <BrowserRouter>
        <Routes>
            <Route path="/" element={<App/>}>
                <Route path="/home" element={<Home/>}></Route>
                <Route path="/about" element={<About/>}></Route>
            </Route>
        </Routes>
    </BrowserRouter>
)
export default baseRouter
```

【3、替换顶级组件】在/src/main.tsx中把顶级组件**App替换为这个路由对象**：

```
// 引入路由对象
import Router from './router';

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <Router />
  </React.StrictMode>
)
```

【4、添加窗口组件】/src/App.tsx中，使用 组件作为占位符组件：

```
import {Outlet} from "react-router-dom";
function App() {
  //const [count, setCount] = useState(0)

  return (
    <div className="App">
      {/* 占位符组件，窗口，有点类似于Vue中的 router-view */}
      <Outlet />
    </div>
  )
}

export default App
```

这样就可以在浏览器中访问，下面的地址看到页面了：

```
http://localhost:3002/home
http://localhost:3002/about
```

9.2、编程式导航--设置菜单点击跳转

/src/App.tsx中，使用 组件进行跳转：

```
import {Outlet,Link} from "react-router-dom"
...
<div className="App">
    <Link to="/home">home</Link> |
    <Link to="/about">about</Link>
    {/* 占位符组件，窗口，有点类似于Vue中的 router-view */}
    <Outlet />
</div>
```

9.3、配置重定向

/src/router/index.tsx中：

```
import {BrowserRouter,Routes,Route,Navigate} from "react-router-dom"
...
<Route path="/" element={<App/>}>
    {/* 配置 用户访问/的时候，重定向到/home路径 */}
    <Route path="/" element={<Navigate  to="/home" />}></Route>
    ...
</Route>
```

十、React路由——第二种配置方案

10.1、路由表的写法

和上面一样，在这里模拟vue中的home和about两个组件展示

【1、准备界面】在上面已经完成

【2、配置对应关系】/src/router/index.tsx中：

```
import Home from "../views/Home"
import About from "../views/About"

// Navigate 重定向组件
import {Navigate} from "react-router-dom"


const routes = [
    {
        path:"/",    //重定向到home
        element:<Navigate  to="/home" />,
    },
    {
        path:"/home",
        element:<Home/>,
    },
    {
        path:"/about",
        element:<About/>,
    }
    // { path: "*", element: <Navigate  to="/" /> },
]

export default routes
```

【3、路由组件的添加】在/src/main.tsx中在顶级组件**App外层添加路由组件**：

```
...
// 路由
import { BrowserRouter } from "react-router-dom"
import App from "./App"

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
)
```

【4、添加窗口组件】/src/App.tsx中，使用hooks来创建占位符：

```
import {useRoutes,Link} from "react-router-dom"
import router from './router';

function App() {
  const outlet = useRoutes(router);
  return (
    <div className="App">
      <Link to="/home">home2</Link> |
      <Link to="/about">about2</Link>
       {/* 占位符组件，窗口，有点类似于Vue中的 router-view */}
      {outlet}
    </div>
  )
}
export default App
```

10.2、路由懒加载

/src/router/index.tsx中把About做成懒加载组件：

```
import { lazy } from 'react'
import Home from "../views/Home"
const About = lazy(() => import("../views/About"))
```

10.3、懒加载组件需要嵌套Loading组件的报错解决

报错解决：此时lazy懒加载的话还是会报错

```
▶The above error occurred in the <Route.Provider> component:                react-dom.

    at Outlet (http://localhost:2020/node_modules/.vite/deps/react-router-dom.js?v=7d56e00e:1060:26)
    at main
    at http://localhost:2020/node_modules/.vite/deps/chunk-3SGPN2FI.js?v=7d56e00e:1588:50
    at div
    at http://localhost:2020/node_modules/.vite/deps/chunk-3SGPN2FI.js?v=7d56e00e:1588:50
    at Box3 (http://localhost:2020/node_modules/.vite/deps/chunk-3SGPN2FI.js?v=7d56e00e:6688:19)
    at PersistentDrawerLeft (http://localhost:2020/src/views/Home.tsx?t=1653496389623:97:22)
    at div
    at App (http://localhost:2020/src/App.tsx?t=1653497719606:21:18)
    at Router (http://localhost:2020/node_modules/.vite/deps/react-router-dom.js?v=7d56e00e:1067:15)
    at BrowserRouter (http://localhost:2020/node_modules/.vite/deps/react-router-dom.js?v=7d56e00e:1196:5)

Consider adding an error boundary to your tree to customize error handling behavior.
Visit https://reactjs.org/link/error-boundaries to learn more about error boundaries.
▶Uncaught Error: A component suspended while responding to synchronous input. This will cause the UI to be react-dom.
replaced with a loading indicator. To fix, updates that suspend should be wrapped with startTransition.
    at throwException (react-dom.development.js:18940:35)
    at handleError (react-dom.development.js:26267:7)
    at renderRootSync (react-dom.development.js:26393:7)
    at recoverFromConcurrentError (react-dom.development.js:25806:20)
    at performSyncWorkOnRoot (react-dom.development.js:26052:20)
    at flushSyncCallbacks (react-dom.development.js:12009:22)
    at react-dom.development.js:25607:13
```

```
Uncaught Error: A component suspended while responding to synchronous input.
This will cause the UI to be replaced with a loading indicator.
```

React路由懒加载的在嵌套路由中的时候强制要求，展示时候必须要有loading组件

/src/router/index.tsx中：

```
{
    path:"/about",
        element:
        // 这种写法是来解决下面这个报错的
            <React.Suspense fallback={<div>Loading...</div>}>
            <About />
        </React.Suspense>
},
```

简化后的写法

```
import React from 'react';
import { lazy } from 'react'
import Home from "../views/Home"
const About = lazy(() => import("../views/About"))
// Navigate 重定向组件
import {Navigate} from "react-router-dom"
const withLoadingComponent = (comp:JSX.Element)=>{
    return <React.Suspense fallback={<div>Loading...</div>}>
            {comp}
        </React.Suspense>
}
const routes = [
    {
        path:"/",    //重定向到home
        element:<Navigate  to="/home" />,
    },
    {
        path:"/home",
        element:<Home/>,
    },
    {
        path:"/about",
```

```
        element:withLoadingComponent(<About/>),
    }
    // { path: "*", element: <Navigate  to="/" /> },
]
export default routes
```

十一、布局解决方案——Layout组件（含侧边栏）

布局组件文档：https://ant.design/components/layout-cn/

注意：不要直接点复制代码，先展开代码部分，里面也包含了样式代码。如果直接点复制代码，样式代码直接被忽略！

11.1、初步使用

App组件中：

```
import {useRoutes} from "react-router-dom"
import router from "./router"
function App() {
  const outlet = useRoutes(router)
  return (
    <div className="App">
      {outlet}
    </div>
  )
}
export default App
```

Home.tsx组件中：

```
import {
    DesktopOutlined,
    FileOutlined,
    PieChartOutlined,
    TeamOutlined,
    UserOutlined,
  } from '@ant-design/icons';
  import type { MenuProps } from 'antd';
  import { Breadcrumb, Layout, Menu } from 'antd';
  import React, { useState } from 'react';

  const { Header, Content, Footer, Sider } = Layout;

  type MenuItem = Required<MenuProps>['items'][number];

  function getItem(
    label: React.ReactNode,
    key: React.Key,
    icon?: React.ReactNode,
    children?: MenuItem[],
  ): MenuItem {
    return {
      key,
      icon,
      children,
```

```
      label,
    } as MenuItem;
  }

  const items: MenuItem[] = [
    getItem('Option 1', '1', <PieChartOutlined />),
    getItem('Option 2', '2', <DesktopOutlined />),
    getItem('User', 'sub1', <UserOutlined />, [
      getItem('Tom', '3'),
      getItem('Bill', '4'),
      getItem('Alex', '5'),
    ]),
    getItem('Team', 'sub2', <TeamOutlined />, [getItem('Team 1', '6'),
getItem('Team 2', '8')]),
    getItem('Files', '9', <FileOutlined />),
  ];

  const View: React.FC = () => {
    const [collapsed, setCollapsed] = useState(false);

    return (
      <Layout style={{ minHeight: '100vh' }}>
        <Sider collapsible collapsed={collapsed} onCollapse={value =>
setCollapsed(value)}>
          <div className="logo" ></div>
          <Menu theme="dark" defaultSelectedKeys={['1']} mode="inline" items=
{items} />
        </Sider>
        <Layout className="site-layout">
          <Header className="site-layout-background" style={{ padding: 0 }} />
          <Content style={{ margin: '0 16px' }}>
            <Breadcrumb style={{ margin: '16px 0' }}>
              <Breadcrumb.Item>User</Breadcrumb.Item>
              <Breadcrumb.Item>Bill</Breadcrumb.Item>
            </Breadcrumb>
            <div className="site-layout-background" style={{ padding: 24,
minHeight: 360 }}>
              Bill is a cat.
            </div>
          </Content>
          <Footer style={{ textAlign: 'center' }}>Ant Design ©2018 Created by
Ant UED</Footer>
        </Layout>
      </Layout>
    );
  };

  export default View;
```

global.scss添加全局样式

```
.logo {
    height: 32px;
    margin: 16px;
    background: rgba(255, 255, 255, 0.3);
}

.site-layout .site-layout-background {
    background: #fff;
}
```

11.2、右侧样式调整

Home.tsx中：

```
<Layout style={{ minHeight: '100vh' }}>
        {/* 侧边栏 */}
        <Sider collapsible collapsed={collapsed} onCollapse={value =>
setCollapsed(value)}>
            {/* 侧边栏顶部logo */}
            <div className="logo" ></div>
            <Menu theme="dark" defaultSelectedKeys={['1']} mode="inline" items=
{items} />
        </Sider>
        {/* 右侧界面 */}
        <Layout className="site-layout">
            <Header className="site-layout-background" style={{ paddingLeft:
'16px'}} >
                {/* 面包屑 */}
                <Breadcrumb className='crumb'>
                    <Breadcrumb.Item>User</Breadcrumb.Item>
                    <Breadcrumb.Item>Bill</Breadcrumb.Item>
                </Breadcrumb>
            </Header>
             {/* 内容部分 */}
            <Content style={{ margin: '16px' }}>
                <div className="site-layout-background" style={{ padding: 24,
minHeight: 360 }}>
                Bill is a cat.
                </div>
            </Content>
            {/* 页脚部分 */}
          <Footer style={{ textAlign: 'center' }}>Ant Design ©2018 Created by
Ant UED</Footer>
        </Layout>
      </Layout>
```

global.scss样式补充

```
.site-layout .crumb{
    line-height: 64px;
}
```

设置右侧大盒子自动占满

```
<Content style={{ margin: '16px 16px 0' }} className="site-layout-background">
                <div  style={{ padding: 24, minHeight: 360 }}>
                Bill is a cat.
                </div>
            </Content>
            {/* 页脚部分 */}
          <Footer style={{ textAlign: 'center', height:"48px", padding:0,
lineHeight:"48px",}}>通用后台管理系统 ©2022 Created by 前端乐哥</Footer>
```

十二、侧边栏的点击实现跳转

12.1、侧边栏的点击事件

Home组件中：Menu上添加点击事件：

```
const items: MenuItem[] = [
    getItem('栏目 1', '1', <PieChartOutlined />),
    getItem('栏目 2', 2, <DesktopOutlined />),
    getItem('User', 'sub1', <UserOutlined />, [
      getItem('Tom', '3'),
      getItem('Bill', '4'),
      getItem('Alex', '5'),
    ]),
    getItem('Team', 'sub2', <TeamOutlined />, [getItem('Team 1', '6'),
getItem('Team 2', '8')]),
    getItem('Files', '9', <FileOutlined />),
];

...
const menuClick = (e:{key:string}) =>{
     console.log(e.key); // ！！！【重点】获取点击到的key就是上面的这些数字，所以我们需要
把上面的key换成对应路径
}
return(
    ....
    <Menu theme="dark" defaultSelectedKeys={['1']} mode="inline" items={items}
onClick={menuClick}/>
)
```

【重点】`console.log(e.key);`获取点击到的key就是上面的这些数字，所以我们需要把上面的key换成对应路径

```
const items: MenuItem[] = [
    getItem('栏目 1', '/page1', <PieChartOutlined />),
    getItem('栏目 2', '/page2', <DesktopOutlined />),
    getItem('User', 'sub1', <UserOutlined />, [
      getItem('Tom', '3'),
      getItem('Bill', '4'),
      getItem('Alex', '5'),
    ]),
    getItem('Team', 'sub2', <TeamOutlined />, [getItem('Team 1', '6'),
getItem('Team 2', '8')]),
    getItem('Files', '9', <FileOutlined />),
];
```

12.2、配置点击跳转和占位符的展示窗口

Home组件中：

```
import {Outlet,useNavigate} from "react-router-dom"

const View: React.FC = () => {
    ...
    {/* 【!!!!】定义跳转对象  */}
    const navigateTo = useNavigate();
    const menuClick = (e:{key:string}) =>{
        {/* console.log(e.key);*/}
        navigateTo(e.key)
    }

    return(
        ...
        {/* 内容部分 */}
        <Content style={{ margin: '16px 16px 0' }} className="site-layout-
background">
             {/* 【!!!!重点】设置占位符展示窗口
             注意：嵌套路由的占位符展示窗口需要用Outlet组件，这里和根路由的展示有所区别
             */}
            <Outlet />
        </Content>
        ...
    )
}
```

此时还需要配置路由才能实现点击跳转

2.3、嵌套路由的配置

路由中还需要配置，在router/index.tsx 中进行修改：（记得在views中准备好Page1和Page2两个组件）

```
import React,{ lazy } from "react"
// Navigate重定向组件
import {Navigate} from "react-router-dom"

import Home from  "../views/Home"
const Page1 = lazy(()=>import("../views/Page1"))
const Page2 = lazy(()=>import("../views/Page2"))

const withLoadingComponent = (comp:JSX.Element) => (
  <React.Suspense fallback={<div>Loading...</div>}>
    {comp}
  </React.Suspense>
)

const routes = [
  //  嵌套路由 开始--------------------
  {
    path:"/",
    element:<Navigate to="/page1"/>
  },
```

```
  {
    path:"/",
    element: <Home />,
    children:[
      {
        path:"/page1",
        element: withLoadingComponent(<Page1 />)
      },
      {
        path:"/page2",
        element: withLoadingComponent(<Page2 />)
      }
    ]
  }
  //  嵌套路由 结束--------------------
  // {
  //   path:"/home",
  //   element: <Home />
  // },
  // {
  //   path:"/about",
  //   element: withLoadingComponent(<About />)

  // },
  // {
  //   path:"/user",
  //   element: withLoadingComponent(<User />)
  // }
]

export default routes
```

了解来自React18的警告

打开控制台，如果看到如下警告：

```
/*
react-dom.development.js:86 Warning: ReactDOM.render is no longer supported in
React 18. Use createRoot instead. Until you switch to the new API, your app will
behave as if it's running React 17. Learn more: https://reactjs.org/link/switch-
to-createroot
*/
```

课件React18已经不再支持ReactDOM.render。使用createRoot来代替。

所以，index.js中，代码改成(才不会有警告)：

```
import * as ReactDOMClient from 'react-dom/client';
import App from './App';


const container = document.getElementById('root');
// Create a root.
const root = ReactDOMClient.createRoot(container);

// Initial render: Render an element to the root.
root.render(<App  />);
```