接口文档  
https://blog.csdn.net/qq\_20503733/article/details/105239905

后端接口运行命令  
node ./app.js

//接口访问本地服务器地址  
http://127.0.0.1:8888/api/private/v1/

如果前端和后端接口之间不存在跨域的问题，这使用cookie和session来记录状态就可以；如果服务器和前端之间存在跨域的问题，这需要用token来维持用户的登录状态

跨域问题及解决方法参考以下链接https://vue3js.cn/interview/vue/cors.html#%E4%B8%80%E3%80%81%E8%B7%A8%E5%9F%9F%E6%98%AF%E4%BB%80%E4%B9%88



# 一、登录模块login.vue

子盒子在父盒子中上下左右居中

position: absolute;  
left: 50%;  
top:50%;  
transform: translate(-50%,-50%);

1.样式中使用less的时候报

Module build failed: TypeError: this.getOptions is not a function at Object.lessLoader

原因：因为less-loader的版本太高，换成5.0.0的即可

2.安装element-ui之后，没有全局引入导致的错误

解决方法：

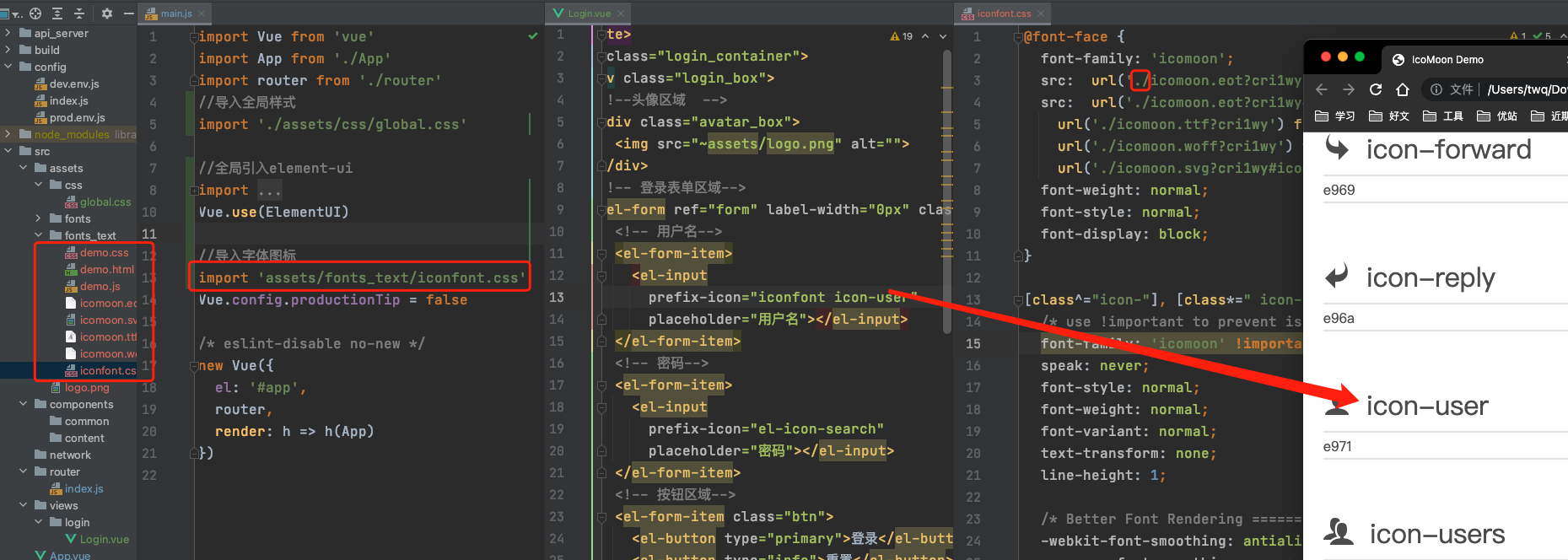
在main.js中全局引入

import ElementUI from 'element-ui'  
import 'element-ui/lib/theme-chalk/index.css'  
Vue.use(ElementUI)

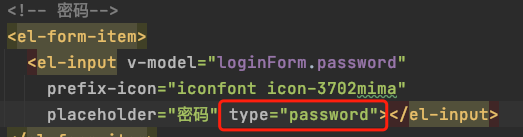
3.解决padding导致的盒子大小改变

box-sizing: border-box;

4.element带字体图标的表单：（将原网站下载下来的icommon文件的css，以及js都放入fonts中，然后按以下方法自定义图标）



5.密码输入框的结果看不到



6．网络请求

request.js

import ***axios*** from "axios";  
export function request(config){  
 const instance = ***axios***.create({  
 //1.创建axios实例  
 baseURL:'http://127.0.0.1:8888/api/private/v1/',  
 timeout:5000,  
 })  
 //2.axios的拦截器  
 instance.interceptors.request.use(config =>{  
 return config  
 },error => {  
 ***console***.log(error);  
 })  
 //3.响应拦截  
 instance.interceptors.response.use(res =>{  
 return res.data  
 },error => {  
 ***console***.log(error);  
 })  
 //4.发送真正的网络请求  
 return instance(config)  
}

7.token值的保存，因为在后面访问接口都需要在登录的前提下才能允许访问



8.如果用户没有登录，但是直接通过URL访问特定的页面，需要重新导航到登录页面，

在路由里配置

//挂载陆游导航守卫  
// 为路由对象，添加 beforeEach 导航守卫  
***router***.beforeEach((to,from,next)=>{  
 //to 将要访问的路径  
 //from 代表从哪个路径跳转而来  
 //next是一个函数，表示放行  
 // next()放行 next('/login')强制跳转  
   
   
 if(to.path === '/login') return next()//如果当前路径是登录界面路径直接放行即可  
 //获取token  
 const tokenstr = ***window***.sessionStorage.getItem('token')  
 if(!tokenstr) return next('/login')//如果用户还处于未登录的状态，则强制跳转到登录界面  
 next()//用户处于登录状态，直接放行  
})

9.退出功能

//清空session值，并且回到登录界面  
***window***.sessionStorage.clear()  
this.$router.push('/login')

10.webstorm格式化

10.1快捷键：cmd + alt + L

10.2 根目录下添加.prettierrc文件，然后写上以下两行

Semi:false表示格式化的时候不需要加分号，singleQuote双引号变单引号

{  
 "semi": false,  
 "singleQuote": true  
}

# 二、.git合并分支并推送

假设当前在login分支上，修改文件之后

git checkout -b hmshop 创建新的分支hmshop 并切换到该分支

git add .

git commit –m ‘2’

git push -u origin hmshop 在远程仓库中创建hmshop分支 将本地代码推送到远程仓库中

git checkout master 切换到master分支

git merge hmshop 在master分支中合并hmshop的代码

git push 将合并好的代码重新推送到远程仓库master分之上

# 三、home

## 3.1侧边栏

1.css操作盒子的排列

justify-content: space-between;//两个元素左右排列(一个靠最左，一个靠最右)  
align-items: center;//上下居中

2.当用户只有在登录的情况下才能进行后面接口数据请求

//2.axios的拦截器  
instance.interceptors.request.use(config =>{  
 //为请求头对象，添加验证的Authorization字段  
 //后面接口的请求，都必须是在登录之后的情况下才能请求  
 config.headers.Authorization = ***window***.sessionStorage.getItem('token')  
 return config  
},error => {  
 ***console***.log(error);  
})

3.一级菜单中每一项都是同一个图标的问题解决方法：

利用每个一级菜单都有一个id，用一个对象存储每个一级菜单的id,及对应的图标，最后用class绑定即可

<!--图标-->  
<i :class="iconsObj[***item***.id]"></i>

iconsObj:{  
 '125':'iconfont icon-users',  
 '103':'iconfont icon-tijikongjian',  
 '101':'iconfont icon-3702mima',  
 '102':'iconfont icon-danju',  
 '145':'iconfont icon-baobiao',  
},

4.保证每次只能有一个一级菜单被打开



5.字体的间隔样式的设置

letter-spacing: 0.2em;

## 3.2用户列表

1.模版插槽



2.分页功能





3.修改用户状态，并在后台数据库中保存修改后的结果

（1）用@change监听用户状态的改变，并且将改变之后的数据传递出去

<el-table-column label="状态">  
 <template slot-scope="scope">  
 <!-- scope.row可以拿到这一行的数据-->  
 <el-switch  
 v-model="scope.row.mg\_state" @change="userStateChanged(scope.row)">  
 </el-switch>  
 </template>  
</el-table-column>

（2）//监听 switch 开关状态的改变  
userStateChanged(userInfo){  
 ***console***.log(userInfo);  
 this.setInfoData(userInfo)  
},

（3）//修改用户状态请求接口  
export function setUserInfo(userInfo){  
 return request({  
 url:`/users/${userInfo.id}/state/${userInfo.mg\_state}`,  
 method:'put'  
 })  
}

（4）setInfoData(userInfo){  
 setUserInfo(userInfo).then(res =>{  
 if(res.meta.status !== 200){  
 //虽然在数据修改失败，但是在页面上却修改成功，所以要取反回到修改之前的状态  
 userInfo.mg\_state = !userInfo.mg\_state  
 return this.$message.error('更新用户状态失败')  
 }  
 this.$message.success('更新用户状态成功！')  
 ***console***.log(res);  
 })  
}

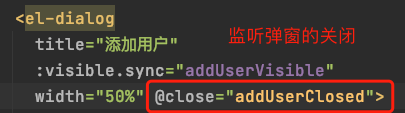
4.用户查询

先实现输入框与query的双向绑定，然后对搜索按钮绑定查询数据接口的函数

<el-input placeholder="请输入内容" v-model="queryInfo.query">  
 <el-button slot="append" icon="el-icon-search" @click="getUserListData"></el-button>  
</el-input>

5.

6.解决关闭对话框，但是上次表单里填写的数据没有清空的问题



当弹窗关闭后重置表单里面的数据

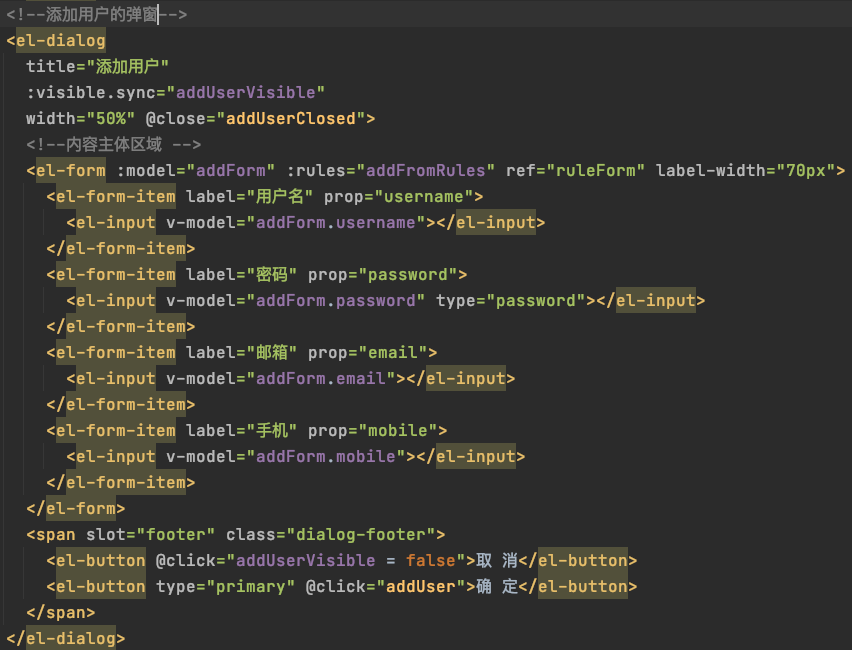


7. vue传参------params和data的区别

params是添加到url的请求字符串中的，用于get请求。而data是添加到请求体（body）中的， 用于post请求。

8.添加用户

（1）写出添加用户弹窗的内容



（2）点击添加用户的时候显示弹窗



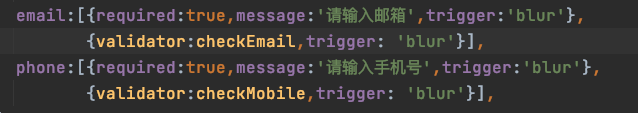
（3）定义表单的验证规则

const regEmail = /^([a**-**zA**-**Z0**-**9\_-])+@([a**-**zA**-**Z0**-**9\_-])+(\.[a**-**zA**-**Z0**-**9\_-])+/

const regMobile = /^(0|86｜17951)?(13[0**-**9]|15[012356789]|17[678]|18[0**-**9]|14[57])[0**-**9]{8}$/



Trigger:表示触发校验的时机，blur表示当失去焦点时进行校验



（4）当用户信息输入无误，点击确定添加



校验表单信息是否符合定义的规则，





9.修改用户

（1）写出要编辑的用户弹窗



（2）点击编辑用户的按钮的时候显示弹窗并展示要编辑的用户的信息





（3）编写获取要编辑用户原来的数据的接口



（4）修改好的数据提交：在弹窗中点击确定表示要提交修改好的用户信息





10.删除用户

（1）根据用户id删除用户



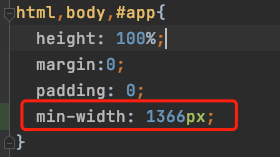


（2）删除请求的编写



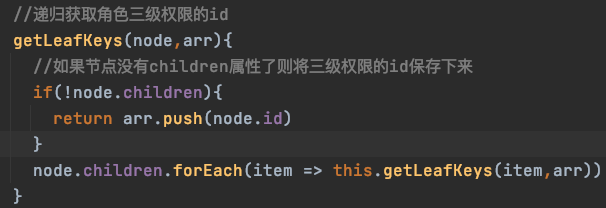
## 3.3角色列表

1.给界面添加最小宽度，防止页面布局改变



2.获取角色三级权限id

(1)自定义递归函数获取三级标题的id



(2)点击分配权限按钮之前调用递归函数，获取角色三级权限的id

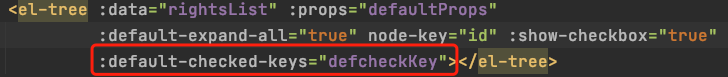
这里通过作用于插槽获取角色



(3)



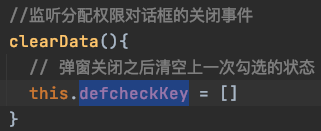
(4)得到三级权限的id后



(5)由此方法衍生出一个bug,就是不同角色三级权限的id在弹窗关闭后没有清空defcheckKey这个数组，因此会导致id冲突

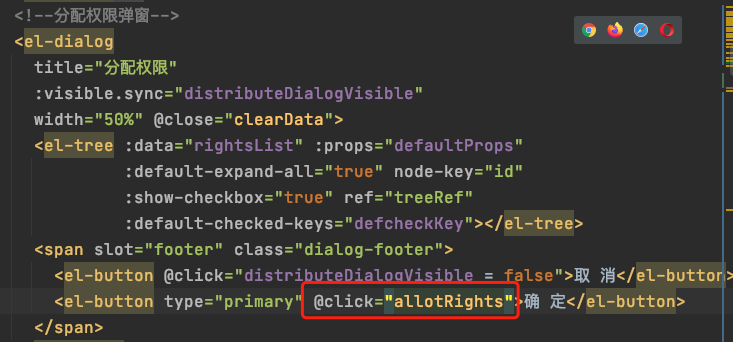
解决方法：监听弹窗关闭事件





3.角色授权

(1)定义点击事件，但点击确定的时候



（2）获取tree中处于选中状态权限的id以及半选中权限状态的id



（3）定义接口



（4）请求添加角色权限

