<template>

<div id="app">

<router-view></router-view>

</div>

</template>

<script>

export default {

name: 'App',

created() {

if(sessionStorage.getItem('state'))

{

this.$store.replaceState(Object.assign({},this.$store.state,JSON.parse(sessionStorage.getItem('state'))));

}

window.addEventListener('beforeunload',()=>{

sessionStorage.setItem('state',JSON.stringify(this.$store.state))

})

}

}

</script>

<style>

</style>

import { createApp } from 'vue'

import App from './App.vue'

import router from "@/router/index.js";

import axiosInstance from "@/api";

import store from "@/store";

import '@imengyu/vue3-context-menu/lib/vue3-context-menu.css'

import ContextMenu from '@imengyu/vue3-context-menu'

const app = createApp(App)

app.use(store)

app.use(router)

app.use(ContextMenu)

app.config.globalProperties.$axios = axiosInstance;

app.mount('#app')

import axiosInstance from "@/api/index";

import {blobToBase64} from "@/utils/BlobData"

import {access} from "@babel/core/lib/config/validation/option-assertions";

const axios = axiosInstance

export const getVercodeimg = () => {

return axios.get('/api/captcha/', {

responseType: 'arraybuffer'

});

}

export const getEmailCode = (email) => {

return axios.get(`/api/emcode/?email=${email}`)

}

export const registerRequest = (user, pwd, email, valid\_code) => {

return axios.post('/api/register/', {

user: user,

pwd: pwd,

email: email,

valid\_code: valid\_code

})

}

export const getUserList = (nid, startIndex, pageSize, query, admin) => {

return axios.post(

'/api/admin/getuser/',

{

nid: nid,

startIndex: startIndex, // 起始索引

pageSize: pageSize, // 每页条数,

query: query,

admin: admin

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const submit360 = (nid, data, describe) => {

return axios.post('/api/submit/360/', {

nid: nid,

data: data,

describe: describe

}, {

headers: {

Accept: 'application/json',

'Content-Type': 'application/json'

}

})

}

export const get360List = (page, pageSize) => {

return axios.get(`/api/get/360/?page=${page}&pageSize=${pageSize}`)

}

export const get360 = (nid) => {

return axios.get(`/api/get/360/display/?nid=${nid}`)

}

export const changeUser = (nid, changenid, username, state, user\_group, email, is\_new\_pasword, new\_password) => {

return axios.post(

'/api/admin/changeuser/',

{

nid: nid,

changenid: changenid,

username: username,

state: state,

user\_group: user\_group,

email: email,

is\_new\_pasword: is\_new\_pasword,

new\_password: new\_password,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const deleteUser = (nid, selected\_nids) => {

return axios.post(

'/api/admin/delete/user/',

{

nid: nid,

selected\_nids: selected\_nids,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const delete3D = (nid, id) => {

return axios.post(

'/api/admin/delete/3D/',

{

nid: nid,

id: id,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const getTagList = (nid) => {

return axios.post(

'/api/admin/get/taglist/',

{

nid: nid,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const getArticleList = (nid, startIndex, catrgpryvalue, tagvalue, query, status) => {

return axios.post(

'/api/admin/get/articlelist/',

{

nid: nid,

startIndex: startIndex, // 起始索引

catrgpryvalue: catrgpryvalue,

tagvalue: tagvalue,

query: query,

status: status

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const getInfo = (nid) => {

return axios.post(

'/api/get/info/',

{

nid: nid

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};

export const changeUserInfo = (data) => {

return axios.post(

'/api/people/change/',

{

data: data

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

}

);

};import Axios from 'axios'

const axiosInstance = Axios.create({

withCredentials:true

})

axiosInstance.interceptors.request.use((config) => {

config.headers['X-Requested-With'] = 'XMLHttpRequest';

const regex = /.\*csrftoken=([^;.]\*).\*$/

config.headers['X-CSRFToken'] = document.cookie.match(regex) === null ? null : document.cookie.match(regex)[1]

return config

})

axiosInstance.interceptors.response.use(

response =>{

return response

},

error => {

return Promise.reject(error)

}

)

export default axiosInstance<template>

<div>

<canvas ref="canvasRef" class="three" id="three"></canvas>

</div>

<div class="content">

<div class="tooltip" id="tooltip">{{ tooltopContent }}</div>

</div>

</template>

<script setup>

import \* as THREE from 'three';

import {OrbitControls} from 'three/examples/jsm/controls/OrbitControls';

import {onMounted, ref} from 'vue';

import {useRoute} from "vue-router";

import {useStore} from "vuex";

import {get360} from "@/api/api";

const canvasRef = ref(null);

let scene;

let camera;

let sphere, controls, renderer, rayCaster;

let curr = 0;

let dataList = [];

let tooltopContent = ref(null); // 展示的内容

let currImage = null;

onMounted(() => {

const store = useStore();

let nid = store.getters.getId

if (nid) {

get360(nid).then(responser => {

dataList = responser.data.data;

switchScene(dataList['spots'][0].image.url, 0)

})

} else {

alert('数据加载失败')

}

})

function initThree() {

scene = new THREE.Scene();

camera = new THREE.PerspectiveCamera(

60,

window.innerWidth / window.innerHeight,

1,

1000

);

rayCaster = new THREE.Raycaster();

camera.position.z = 5;

renderer = new THREE.WebGLRenderer({canvas: canvasRef.value});

renderer.setSize(window.innerWidth, window.innerHeight);

controls = new OrbitControls(camera, renderer.domElement);

controls.rotateSpeed = 0.5;

controls.zoomSpeed = 1.2;

controls.panSpeed = 0.8;

}

function loadModel(url) {

let sphereGeometry = new THREE.SphereGeometry(50, 32, 32);

const texture = new THREE.TextureLoader().load(url);

const material = new THREE.MeshBasicMaterial({

map: texture,

side: THREE.DoubleSide,

transparent: true,

});

sphere = new THREE.Mesh(sphereGeometry, material);

scene.add(sphere)

}

function animate() {

requestAnimationFrame(animate);

renderer.render(scene, camera);

controls.update();

}

function switchScene(url, index) {

canvasRef.value.removeEventListener('click', onClick);

canvasRef.value.removeEventListener('mousemove', onMousemove, false);

currImage = '/api' + url;

curr = index

scene = null;

sphere = null;

initThree()

console.log(currImage)

loadModel(currImage);

animate()

for (let i = 0; i < dataList['spots'][curr].tips.length; i++) {

changeAddSprite(dataList['spots'][curr]['tips'][i])

}

canvasRef.value.addEventListener('click', onClick);

canvasRef.value.addEventListener('mousemove', onMousemove, false);

}

function onClick(event) {

event.preventDefault();

let x = ((event.clientX - document.getElementById("three").getBoundingClientRect().left) / document.getElementById("three").offsetWidth) \* 2 - 1;

let y = -((event.clientY - document.getElementById("three").getBoundingClientRect().top) / document.getElementById("three").offsetHeight) \* 2 + 1;

let standardVector = new THREE.Vector3(x, y, 1);// 标准设备坐标

let worldVector = standardVector.unproject(camera);

let ray = worldVector.sub(camera.position).normalize();

rayCaster.set(camera.position, ray);

const objs = rayCaster.intersectObjects(sphere.children, true);

if (objs.length > 0) {

const spriteClick = objs[0].object;

let index = spriteClick.content.next

if (index !== -1) {

switchScene(dataList['spots'][index].image.url, index)

}

}

}

function changeAddSprite(data) {

let spriteMap = new THREE.TextureLoader().load('/api' + data.image.url);

spriteMap.needsUpdate = true

let spriteMaterial = new THREE.SpriteMaterial({

map: spriteMap,

transparent: true,

alphaTest: 0.5,

});

let newsprite = new THREE.Sprite(spriteMaterial);

newsprite.name = data.tip

newsprite.position.copy(data.position);

newsprite.scale.set(5, 5, 1)

newsprite.center = new THREE.Vector2(0.5, 0)

newsprite.content = data.content;

sphere.add(newsprite);

}

function onMousemove(e) {

let three = document.getElementById("three")

let x = ((e.clientX - three.getBoundingClientRect().left) / three.offsetWidth) \* 2 - 1;

let y = -((e.clientY - three.getBoundingClientRect().top) / three.offsetHeight) \* 2 + 1;

let mouse = new THREE.Vector2(x, y);

rayCaster.setFromCamera(mouse, camera)

let intersects = rayCaster.intersectObjects(sphere.children, true);

let tooltip = document.getElementById('tooltip');

if (intersects.length > 0) {

let p = intersects[0].object.position.clone().project(camera);

if (intersects[0].object.content.showTip) {

tooltopContent.value = intersects[0].object.content.text;

tooltip.style.top = ((-1 \* p.y + 1) \* three.offsetHeight / 2) + 'px';

tooltip.style.left = ((p.x + 1) \* three.offsetWidth / 2) + 'px'

}

} else {

tooltip.style.top = '0';

tooltip.style.left = '0'

tooltopContent.value = null

}

}

</script>

<style scoped>

.three {

width: 100%;

height: 100vh;

display: block;

}

.content {

font-family: serif;

color: rgba(0, 0, 0, .77);

}

.tooltip {

position: absolute;

top: 0;

left: 0;

padding: 10px;

transform: translate3d(-70%, -110%, 0);

transition: opacity .3s;

background-color: rgba(0, 0, 0, 0.8); /\* 使用rgba来设置黑色背景和透明度 \*/

box-shadow: 0 0 10px rgba(0, 0, 0, 0.5); /\* 添加阴影效果 \*/

color: #fff; /\* 设置文本颜色为白色，以确保文本在黑色背景上可见 \*/

}

</style><template>

<el-menu

:default-active="this.$router.path"

class="el-menu-demo"

mode="horizontal"

:ellipsis="false"

@select="handleSelect"

router

>

<el-menu-item index="/">

<el-image class="logo" src="/logo.png"></el-image>

</el-menu-item>

<el-menu-item index="/360">3D展馆</el-menu-item>

<div class="flex-grow"/>

<el-menu-item v-if="getLogin" index="/login">登录</el-menu-item>

<el-menu-item v-if="getLogin" index="/register">注册</el-menu-item>

<el-menu-item>

<el-dropdown :hide-on-click="true" :disabled="getLogin" @command="dropdownFun">

<span class="el-dropdown-link">

<el-avatar v-bind:src="'/api/'+getAvatar"></el-avatar>

<el-text>{{ getUsername }}</el-text>

</span>

<template #dropdown>

<el-dropdown-menu router style="color: #e0e0e0">

<el-dropdown-item :command="'people'">个人中心</el-dropdown-item>

<el-dropdown-item :command="'editor'">系统信息</el-dropdown-item>

<el-dropdown-item :command="'admin'">后台管理</el-dropdown-item>

<el-dropdown-item :command="'logout'">注销</el-dropdown-item>

</el-dropdown-menu>

</template>

</el-dropdown>

</el-menu-item>

</el-menu>

</template>

<script>

import {ElMessage, ElMessageBox} from "element-plus";

import router from "@/router";

export default {

data() {

return {

activeIndex: '1',

}

},

methods: {

handleSelect(key, keyPath) {

},

dropdownFun(comment) {

if (comment === 'logout') {

this.exit()

} else if (comment === 'editor') {

router.push('/editor')

} else if (comment === 'admin') {

router.push('/adminmanage')

} else if (comment === 'people') {

router.push('/people')

}

},

exit() {

ElMessageBox.confirm('此操作将注销登录, 是否继续?', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning'

}).then(() => {

if (sessionStorage.getItem('state')) {

sessionStorage.removeItem('state');

}

this.$router.push('/');

this.$store.commit("logout")

ElMessage({

type: 'success',

message: "退出成功"

})

}).catch(() => {

ElMessage({

type: "info",

message: "已取消操作"

})

})

}

},

computed: {

getAvatar() {

return this.$store.state.user.user.avatar;

},

getUsername() {

return this.$store.state.user.user.username;

},

getLogin() {

return !this.$store.state.user.user.is\_login;

}

}

}

</script>

<style scoped>

.el-menu-demo {

background: #56000B;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.el-menu--horizontal > .el-menu-item {

color: #e0e0e0;

}

.flex-grow {

flex-grow: 1;

}

:deep(:focus) {

outline: none;

}

.logo {

width: 64px;

height: 64px;

}

</style>

<template>

<div>

<div>

<el-card shadow="always" class="card">

<div class="card-headel">

<span>全景管理</span>

</div>

<div class="card-tools">

<div class="tools-btn">

<button class="tools-dele-btn" @click="handleAdd">

<el-icon>

<Plus/>

</el-icon>

<span>添加</span>

</button>

</div>

<div class="select">

<el-select v-model="catrgpryvalue" class="m-2" placeholder="文章分类" @change="handleSelect">

<el-option

v-for="item in options"

:key="item.value"

:label="item.label"

:value="item.value"

/>

</el-select>

<el-select v-model="tagvalue" class="m-2" placeholder="文章标签" @change="handleSelect">

<el-option

v-for="item in optionsTag"

:key="item.value"

:label="item.label"

:value="item.value"

/>

</el-select>

<el-select v-model="blogstatusvalue" class="m-2" placeholder="文章状态" @change="handleSelect">

<el-option

v-for="item in blogstatus"

:key="item.value"

:label="item.label"

:value="item.value"

/>

</el-select>

</div>

<div class="tools-search">

<el-input

v-model="state"

popper-class="my-autocomplete"

placeholder="输入关键字"

@keydown.enter.native="handleSelect"

>

<template #suffix>

<el-icon class="el-input\_\_icon" @click="handleSelect">

<Search/>

</el-icon>

</template>

</el-input>

</div>

</div>

<div class="card-user-list">

<el-card class="card-list" shadow="never">

<ul class="infinite-list" style="overflow: auto">

<li v-for="item in userData" :key="item.nid"

class="infinite-list-item">

<div class="active">

<div class="active-left">

<el-image :src="'/api/'+item.url" class="active-left-img"></el-image>

</div>

<div class="active-right">

<div class="title" v-html="item.title"></div>

<div class="time">{{ formatTimestamp(item.update\_time) }}</div>

<div class="content">

</div>

<div>

<div class="address">

<span>{{ item.province }}</span>

<span>-</span>

<span>{{ item.city }}</span>

<span>-</span>

<span>{{ item.county }}</span>

</div>

<div class="active-tools">

<ElButton text @click="handEditor(item.nid)">编辑</ElButton>

<ElButton text @click="handDisplay(item.nid,item.title)">浏览</ElButton>

<ElButton text @click="handDelete(item.nid)">删除</ElButton>

</div>

</div>

</div>

<el-divider/>

</div>

</li>

</ul>

<div class="pagination-container">

<el-pagination

background layout="prev, pager, next,total"

:current-page="currentPage"

:total="total"

@current-change="handlePageChange"

></el-pagination>

</div>

</el-card>

</div>

</el-card>

</div>

</div>

</template>

<script lang="jsx">

import {Plus, Search} from "@element-plus/icons-vue";

import {onMounted, ref} from "vue";

import {delete3D, get360List, getTagList} from "@/api/api";

import {useStore} from "vuex";

import {ElMessage} from "element-plus";

import {formatTimestamp} from "@/utils/BlobData";

import router from "@/router";

export default {

components: {Plus, Search},

data() {

return {};

},

setup() {

const state = ref("");

const store = useStore();

const userData = ref([]);

const filteredUserData = ref([]);

const catrgpryvalue = ref("不限类型");

const tagvalue = ref("不限类型");

const currentPage = ref(1); // 当前页数

const total = ref(0); // 总条数

const optionsTag = ref([])

const blogstatusvalue = ref("不限类型")

const blogstatus = [

{

value: '不限类型',

label: '不限类型',

},

{

value: '已发布',

label: '已发布',

},

{

value: '草稿',

label: '草稿',

}

]

const options = [

{

value: '不限类型',

label: '不限类型',

},

{

value: '首页',

label: '首页',

},

{

value: '党馆文献',

label: '党馆文献',

},

{

value: '红色影像',

label: '红色影像',

},

{

value: '英雄故事',

label: '英雄故事',

},

]

const handlePageChange = (newPage) => {

currentPage.value = newPage;

loadPanoramaData(); // 在页码变更时重新加载数据

};

const loadPanoramaData = () => {

var nid = store.getters.getUser.nid;

const startIndex = (currentPage.value - 1) \* 10;

get360List(startIndex, 10).then(response => {

if (response.data.msg == 'succeed') {

userData.value = response.data.data

total.value = response.data.total;

} else {

ElMessage.error(response.data.msg)

}

}).catch(() => {

ElMessage.error('数据加载失败')

})

};

const loadTagList = () => {

var nid = store.getters.getUser.nid;

getTagList(nid).then(response => {

if (response.data.msg == '获取成功') {

optionsTag.value = response.data.tags.map(tag => ({value: tag.title, label: tag.title}));

} else {

ElMessage.error('标签列表获取失败');

}

}).catch(error => {

ElMessage.error('标签列表获取失败');

})

}

onMounted(() => {

loadPanoramaData(); // 初始化加载数据

loadTagList();// 初始化加载标签

});

return {

state,

userData,

currentPage,

total,

options,

optionsTag,

handlePageChange,

store,

filteredUserData,

loadPanoramaData,

catrgpryvalue,

tagvalue,

formatTimestamp,

blogstatusvalue,

blogstatus

};

},

methods: {

handleSelect() {

this.loadPanoramaData();

},

handDelete(id) {

delete3D(this.store.getters.getUser.nid, id).then(response => {

if (response.data.message == '删除成功') {

ElMessage.success(response.data.message);

} else

ElMessage.warning(response.data.message);

this.loadPanoramaData()

})

},

handDisplay(nid, title) {

this.$store.commit('updataId', nid);

router.push({

name: 'display',

params: {

username: title

}

})

},

handEditor(nid) {

router.push({

name: '3D',

query: {

id: nid

}

})

},

handleAdd() {

router.push({

name: '3D',

})

}

},

};

</script>

<style scoped>

.card {

flex: 1;

height: 100%;

}

.card-headel {

color: #409eff;

font-family: Raleway, serif;

font-size: 21px;

font-weight: 500;

padding-top: 10px;

padding-bottom: 20px;

line-height: 30px;

}

.select > .el-select {

margin-left: 20px;

}

.card-tools {

font-size: 12px;

display: flex;

}

.card-user-list {

margin-top: 20px;

}

.card-list {

height: 100%;

}

.tools-btn {

float: left;

}

.tools-dele-btn {

padding: 6px 12px;

border: 1px solid #dcdfe6;

color: #fff;

background-color: #f56c6c;

line-height: 1;

white-space: nowrap;

}

.tools-search {

margin-left: 20px;

float: left;

}

.pagination-container {

display: flex;

justify-content: center; /\* 水平居中 \*/

align-items: center; /\* 垂直居中 \*/

margin-top: 20px;

}

.active {

position: relative;

width: calc(100% - 100px);

height: 150px;

margin: 20px;

border-radius: 10px;

}

.active-left {

position: absolute;

top: 10px;

left: 10px;

display: inline-block;

width: 230px;

height: 130px;

border-radius: 10px;

background-color: lightgray;

}

.active-left-img {

width: 100%;

height: 100%;

}

.active-right {

position: absolute;

top: 10px;

right: 10px;

display: inline-block;

width: calc(100% - 260px);

height: 130px;

}

.title {

width: calc(100% - 100px);

float: left;

font-size: x-large;

}

.time {

width: 100px;

float: right;

text-align: right;

margin-top: 10px;

white-space: nowrap;

}

.content {

float: left;

margin-top: 10px;

word-break: break-all;

width: 100%;

height: 60px;

font-size: 12px;

align-items: center;

-webkit-box-align: center;

display: -webkit-box;

-webkit-box-orient: vertical;

-webkit-line-clamp: 3;

overflow: hidden;

background-color: #fff;

}

.content > span {

margin-top: 10px;

color: #ff6666;

background-color: #fff5f2;

}

.active-tools {

float: right;

}

.active-tools > ElButton {

color: #222226;

padding: 0 12px;

cursor: pointer;

margin: 0;

font-size: 14px;

}

.infinite-list-item {

list-style-type: none;

padding: 0;

}

.infinite-list {

padding: 0;

}

</style>

<template>

<div>

<div>

<el-card shadow="always" class="card">

<div class="card-headel">

<span>用户管理</span>

</div>

<div class="card-tools">

<div class="tools-btn">

<button class="tools-dele-btn" @click="handleDeleteSelected">

<el-icon>

<Delete/>

</el-icon>

<span>删除</span>

</button>

</div>

<div class="tools-search">

<el-input

v-model="state"

popper-class="my-autocomplete"

placeholder="输入关键字"

@keydown.enter.native="handleSearch"

>

<template #suffix>

<el-icon class="el-input\_\_icon" @click="handleIconClick">

<Search/>

</el-icon>

</template>

</el-input>

</div>

<div class="tools-switch">

<el-switch

v-model="switchvalue"

class="mb-2"

active-text="仅显示管理员"

inactive-text="全部"

@change="toggleSwitch"

/>

</div>

</div>

<div class="card-user-list">

<el-card class="card-list" shadow="never">

<el-table-v2

:columns="columns"

:data="filteredUserData"

:row-height="50"

:width="1190"

:height="tableHeight"

@selection-change="handleSelectionChange"

>

<template #empty>

<div class="flex items-center justify-center h-100%">

<el-empty/>

</div>

</template>

</el-table-v2>

<div class="pagination-container">

<el-pagination

background layout="prev, pager, next,sizes,total"

:current-page="currentPage"

:page-sizes="[10, 20, 50]"

:page-size="pageSize"

:total="total"

@size-change="handleSizeChange"

@current-change="handlePageChange"

></el-pagination>

</div>

</el-card>

</div>

</el-card>

<el-dialog

v-model="dialogVisible"

title="用户"

width="30%"

:before-close="handleClose"

>

<el-form

ref="ruleFormRef"

label-position="left"

label-width="100px"

:model="formLabelAlign"

:rules="ruleForm"

style="max-width: 460px"

>

<el-form-item label="用户名">

<el-input v-model="formLabelAlign.username"/>

</el-form-item>

<el-form-item label="邮箱">

<el-input v-model="formLabelAlign.email"/>

</el-form-item>

<el-form-item label="设置新密码">

<el-switch

v-model="formLabelAlign.is\_new\_pasword"

class="mb-2"

/>

</el-form-item>

<el-form-item label="新密码" v-if="formLabelAlign.is\_new\_pasword" prop="new\_password">

<el-input v-model="formLabelAlign.new\_password"/>

</el-form-item>

<el-form-item label="用户角色">

<el-select v-model="formLabelAlign.user\_group">

<el-option label="用户" value="customer"/>

<el-option label="普通管理员" value="admin"/>

<el-option label="超级管理员" value="superAdmin"/>

</el-select>

</el-form-item>

<el-form-item label="状态">

<el-switch

v-model="formLabelAlign.state"

class="mb-2"

active-text="封禁"

inactive-text="正常"

/>

</el-form-item>

</el-form>

<template #footer>

<span class="dialog-footer">

<el-button @click="dialogVisible = false">取消</el-button>

<el-button type="primary" @click="handleConfirm">

确定

</el-button>

</span>

</template>

</el-dialog>

</div>

</div>

</template>

<script lang="jsx">

import {Delete, Search, Edit} from "@element-plus/icons-vue";

import {h, nextTick, onMounted, reactive, ref} from "vue";

import {changeUser, deleteUser, getUserList} from "@/api/api";

import {useStore} from "vuex";

import {ElButton, ElMessage, ElMessageBox, ElTag} from "element-plus";

import {formatTimestamp} from "@/utils/BlobData";

export default {

components: {Search, Delete},

data() {

return {

ruleForm: {

new\_password: [

{required: true, message: '密码不能为空', trigger: 'blur'},

{min: 6, max: 20, message: '密码长度在6到20个字符之间', trigger: 'blur'}

],

},

};

},

setup() {

const state = ref("")

const store = useStore();

const selectedRows = ref([]);

const switchvalue = ref(false);

const userData = ref([]);

const columnDataKey = ["username", "email", "create\_time", "user\_group", "is\_state"];

const columnName = ["用户名", "邮箱", "创建时间", "用户角色", "状态", "操作"];

const dialogVisible = ref(false);

const formLabelAlign = reactive({})

const filteredUserData = ref([]);

const tableHeight = ref(550);

const selectedId = ref([]);

const SelectionCell = ({rowData}) => {

const onChange = (value) => (rowData.checked = value);

return (

<ElCheckbox

onChange={onChange}

modelValue={rowData.checked}

indeterminate={rowData.intermediate}

/>

);

};

const columns = [

{

key: 'selection',

dataKey: 'selection',

title: 'selection',

width: 50,

align: 'center',

cellRenderer: ({rowData}) => {

rowData.id = rowData.nid

const onChange = (value) => {

rowData.checked = value;

if (rowData.checked) {

selectedId.value.push(rowData.rowId);

} else {

const idIndex = selectedId.value.findIndex((item) => item === rowData.rowId);

selectedId.value.splice(idIndex, 1);

}

return rowData.checked;

};

return <SelectionCell rowData={rowData} onChange={onChange}/>;

},

headerCellRenderer: () => {

const onChange = (value) => {

const newData = filteredUserData.value.map((row) => ({...row, checked: value}));

filteredUserData.value = newData;

selectedId.value = value ? filteredUserData.value.map((row) => row.rowId) : [];

};

const allSelected =

filteredUserData.value.length > 0 &&

filteredUserData.value.every((row) => row.checked);

return (

<ElCheckbox

onChange={onChange}

modelValue={allSelected}

indeterminate={!allSelected && filteredUserData.value.some((row) => row.checked)}

/>

);

},

},

{

key: columnDataKey[0],

dataKey: columnDataKey[0],

title: columnName[0],

width: 190,

flex: true,

},

{

key: columnDataKey[1],

dataKey: columnDataKey[1],

title: columnName[1],

width: 200,

flex: true,

},

{

key: columnDataKey[2],

dataKey: columnDataKey[2],

title: columnName[2],

width: 200,

cellRenderer: ({cellData}) => {

return formatTimestamp(cellData);

},

},

{

key: columnDataKey[3],

dataKey: columnDataKey[3],

title: columnName[3],

width: 150,

flex: true,

},

{

key: columnDataKey[4],

dataKey: columnDataKey[4],

title: columnName[4],

width: 150,

cellRenderer: ({cellData}) => {

const text = cellData ? "封禁" : "正常";

const type = cellData ? "danger" : "success";

return h(ElTag, {type}, {default: () => text});

},

},

{

key: "edit",

dataKey: "edit",

title: columnName[5],

width: 200,

cellRenderer: (rowData) => (

<>

<ElButton type="primary" onClick={() => handleEdit(rowData)}>

<el-icon><Edit/></el-icon>

</ElButton>

<ElButton type="danger" onClick={() => handleDelete(rowData)}>

<el-icon><Delete/></el-icon>

</ElButton>

</>

),

},

];

const currentPage = ref(1); // 当前页数

const pageSize = ref(10); // 每页显示的条数

const total = ref(0); // 总条数

function handleDelete(rowData) {

ElMessageBox.confirm('确定要删除吗？', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning',

}).then(() => {

deleteUser(store.getters.getUser.nid, [rowData.rowData.nid]).then(response => {

if (response.data.msg == '删除成功') {

ElMessage.success(response.data.msg);

} else {

ElMessage.warning(response.data.error);

}

loadUserData();

}).catch(error => {

ElMessage.error('删除失败，请重试！');

});

}).catch(() => {

});

}

function handleEdit(rowData) {

formLabelAlign.username = rowData.rowData.username;

formLabelAlign.email = rowData.rowData.email;

formLabelAlign.user\_group = rowData.rowData.user\_group;

formLabelAlign.state = rowData.rowData.is\_state;

formLabelAlign.rowIndex = rowData.rowIndex;

formLabelAlign.is\_new\_pasword = false;

formLabelAlign.new\_password = null;

dialogVisible.value = true

}

const handlePageChange = (newPage) => {

currentPage.value = newPage;

loadUserData(); // 在页码变更时重新加载数据

};

const loadUserData = () => {

var nid = store.getters.getUser.nid;

const startIndex = (currentPage.value - 1) \* pageSize.value;

const query = state.value

getUserList(nid, startIndex, pageSize.value, query, switchvalue.value).then((response) => {

userData.value = response.data.data.map((item) => ({...item, rowId: item.nid}));

total.value = response.data.total;

filteredUserData.value = userData.value;

tableHeight.value = pageSize.value \* 50 + 50;

});

};

const handleSelectionChange = (selectedRows) => {

selectedRows.value = selectedRows;

};

onMounted(() => {

loadUserData(); // 初始化加载数据

});

return {

state,

columns,

userData,

currentPage,

pageSize,

total,

handlePageChange,

dialogVisible,

formLabelAlign,

store,

switchvalue,

filteredUserData,

loadUserData,

selectedRows,

handleSelectionChange,

tableHeight

};

},

methods: {

handleSearch() {

this.loadUserData();

},

handleIconClick() {

this.loadUserData();

},

handleDeleteSelected() {

ElMessageBox.confirm('确定要删除吗？', '提示', {

confirmButtonText: '确定',

cancelButtonText: '取消',

type: 'warning',

}).then(() => {

const selectedRowsData = this.filteredUserData.filter((row) => row.checked);

if (selectedRowsData.length > 0) {

const selectedNids = selectedRowsData.map((row) => row.nid);

deleteUser(this.store.getters.getUser.nid, selectedNids).then(response => {

if (response.data.msg == '删除成功') {

ElMessage.success(response.data.msg);

} else

ElMessage.warning(response.data.error);

this.loadUserData()

}).catch(error => {

ElMessage.error('删除失败，请重试！');

});

} else {

ElMessage.warning('请先选中要删除的用户');

}

}).catch(() => {

});

},

toggleSwitch() {

this.loadUserData()

},

handleClose() {

this.dialogVisible = false

},

handleSizeChange(number) {

this.pageSize = number

this.loadUserData()

},

handleConfirm() {

this.$refs.ruleFormRef.validate((valid) => {

if (valid) {

this.userData[this.formLabelAlign.rowIndex].username = this.formLabelAlign.username

this.userData[this.formLabelAlign.rowIndex].is\_state = this.formLabelAlign.state

this.userData[this.formLabelAlign.rowIndex].user\_group = this.formLabelAlign.user\_group

this.userData[this.formLabelAlign.rowIndex].email = this.formLabelAlign.email

changeUser(

this.store.getters.getUser.nid,

this.userData[this.formLabelAlign.rowIndex].nid,

this.formLabelAlign.username,

this.formLabelAlign.state,

this.formLabelAlign.user\_group,

this.formLabelAlign.email,

this.formLabelAlign.is\_new\_pasword,

this.formLabelAlign.new\_password

).then(response => {

alert(response.data.msg)

})

this.dialogVisible = false;

}

})

}

},

};

</script>

<style scoped>

.card {

flex: 1;

height: 100%;

}

.card-headel {

color: #409eff;

font-family: Raleway, serif;

font-size: 21px;

font-weight: 500;

padding-top: 10px;

padding-bottom: 20px;

line-height: 30px;

}

.card-tools {

font-size: 12px;

display: flex;

}

.card-user-list {

margin-top: 20px;

}

.card-list {

height: 100%;

}

.tools-btn {

float: left;

}

.tools-dele-btn {

padding: 6px 12px;

border: 1px solid #dcdfe6;

color: #fff;

background-color: #f56c6c;

line-height: 1;

white-space: nowrap;

}

.tools-search {

margin-left: 20px;

float: left;

}

.tools-switch {

margin-left: 20px;

float: left;

}

.pagination-container {

display: flex;

justify-content: center; /\* 水平居中 \*/

align-items: center; /\* 垂直居中 \*/

margin-top: 20px;

}

</style>

<script>

import {ref, watch} from "vue";

import {More, Plus} from "@element-plus/icons-vue";

export default {

components: {More, Plus},

props: {

info: {

'start': null,

'360': [],

'guidepost': [],

'hint': []

}

},

setup(props,) {

const dropdownCardInfo = ref({

'start': null,

'360': [],

'guidepost': [],

'hint': []

});

const fileList = ref([])

watch(() => props.info, (newVal) => {

dropdownCardInfo.value = newVal

console.log(dropdownCardInfo.value)

}, {immediate: true});

const dialogImageUrl = ref('')

const dialogVisible = ref(false)

return {

dialogImageUrl,

dropdownCardInfo,

fileList,

dialogVisible,

}

},

methods: {

setDropdownCardInfo(type, item) {

let tempData = null;

for (let arrayType in this.dropdownCardInfo) {

if (this.dropdownCardInfo[arrayType] !== null && arrayType !== 'start') {

for (let i = 0; i < this.dropdownCardInfo[arrayType].length; i++) {

if (this.dropdownCardInfo[arrayType][i].uid == item.uid) {

tempData = this.dropdownCardInfo[arrayType][i];

break

}

}

} else if (arrayType == 'start' && this.dropdownCardInfo[arrayType] !== null) {

if (this.dropdownCardInfo[arrayType].uid === item.uid) {

tempData = this.dropdownCardInfo[arrayType]

}

}

if (tempData !== null) break;

}

this.removeItemFromAllArray(item);

if (type === 'start') {

if (this.dropdownCardInfo[type] !== null) {

this.dropdownCardInfo['360'].push(this.dropdownCardInfo[type])

}

this.dropdownCardInfo[type] = tempData;

} else {

this.dropdownCardInfo[type].push(tempData);

}

},

removeItemFromAllArray(item) {

if (this.dropdownCardInfo['start'] !== null && this.dropdownCardInfo['start'].uid === item.uid) {

this.dropdownCardInfo['start'] = null;

}

for (const arrayType in this.dropdownCardInfo) {

if (this.dropdownCardInfo[arrayType] !== null && arrayType !== 'start') {

this.dropdownCardInfo[arrayType] = this.dropdownCardInfo[arrayType].filter(existingItem => existingItem.uid !== item.uid);

}

}

},

handleDeleteSelected() {

for (const item of this.fileList) {

if (item.selected) {

this.removeItemFromAllArray(item)

}

}

this.fileList = this.fileList.filter(item => !item.selected);

},

groupingTips(item) {

let type = null;

for (let arrayType in this.dropdownCardInfo) {

if (this.dropdownCardInfo[arrayType] !== null && arrayType !== 'start') {

for (let i = 0; i < this.dropdownCardInfo[arrayType].length; i++) {

if (this.dropdownCardInfo[arrayType][i].uid == item.uid) {

type = arrayType;

break

}

}

} else if (arrayType == 'start' && this.dropdownCardInfo[arrayType] !== null) {

if (this.dropdownCardInfo[arrayType].uid === item.uid) {

type = arrayType

}

}

if (type !== null) break;

}

if (type == 'start')

return '起点';

else if (type == 'guidepost')

return '路标';

else if (type == 'hint')

return '提示';

else if (type == '360')

return '图片';

else return '无'

},

getBase64(data) {

return new Promise((resolve, reject) => {

const fileReader = new FileReader();

fileReader.onload = (e) => {

resolve(e.target.result);

};

fileReader.readAsDataURL(data);

fileReader.onerror = () => {

reject(new Error('文件流异常'));

};

});

},

handleFileSelect(file, fileList) {

this.getBase64(file.raw).then(response => {

let data = {

name: file.name,

uid: file.uid,

url: file.url,

data: response

}

this.dropdownCardInfo['360'].push(data);

})

},

handleClose() {

this.$emit("update:dialogTableVisible", false);

},

handConfirm() {

this.$emit("update:dialogTableVisible", false);

this.$emit('update:dropdown-card-info', this.dropdownCardInfo)

},

getUrl(url) {

if (url && url.startsWith('/media')) {

return '/api' + url;

} else {

return url;

}

}

}

}

</script>

<template>

<el-dialog title="添加图片"

style="max-height: 860px;overflow-y: hidden" width="70%" :before-close="handleClose">

<div>

<el-row>

<el-upload

v-model:file-list="fileList"

class="upload-demo"

action="https://run.mocky.io/v3/9d059bf9-4660-45f2-925d-ce80ad6c4d15"

list-type="picture"

:auto-upload="false"

:show-file-list="false"

:multiple="true"

@change="handleFileSelect"

>

<el-button type="primary" class="el-upload-button">添加</el-button>

</el-upload>

<el-button type="primary" class="el-upload-button" @click="handleDeleteSelected">删除</el-button>

</el-row>

<div class="scroll-container">

<el-row class="card-row">

<el-card v-if="dropdownCardInfo.start"

:key="dropdownCardInfo.start.uid"

class="box-card">

<div class="image-container">

<el-dropdown trigger="click" class="el-dropdown">

<span class="el-dropdown-link">

<el-icon class="el-icon--right"><More/></el-icon>

</span>

<template #dropdown>

<el-dropdown-item @click="setDropdownCardInfo('start', dropdownCardInfo.start)">起点</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('360', dropdownCardInfo.start)">图片</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('guidepost', dropdownCardInfo.start)">路标</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('hint', dropdownCardInfo.start)">提示</el-dropdown-item>

</template>

</el-dropdown>

<img :src="getUrl(dropdownCardInfo.start.url)" alt="" class="preview-image">

<div class="card-bottom">

<span class="tip">{{ groupingTips(dropdownCardInfo.start) }}</span>

<el-checkbox v-model="dropdownCardInfo.start.selected" class="checkbox"></el-checkbox>

</div>

</div>

</el-card>

<el-card v-for="(item, index) in dropdownCardInfo['360']"

:key="item.uid"

class="box-card">

<div class="image-container">

<el-dropdown trigger="click" class="el-dropdown">

<span class="el-dropdown-link">

<el-icon class="el-icon--right"><More/>

</el-icon>

</span>

<template #dropdown>

<el-dropdown-item @click="setDropdownCardInfo('start', item)">起点</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('360', item)">图片</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('guidepost', item)">路标</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('hint', item)">提示</el-dropdown-item>

</template>

</el-dropdown>

<img :src="getUrl(item.url)" alt="" class="preview-image">

<div class="card-bottom">

<span class="tip">{{ this.groupingTips(item) }}</span>

<el-checkbox v-model="item.selected" class="checkbox"></el-checkbox>

</div>

</div>

</el-card>

<el-card v-for="(item, index) in dropdownCardInfo.guidepost"

:key="item.uid"

class="box-card">

<div class="image-container">

<el-dropdown trigger="click" class="el-dropdown">

<span class="el-dropdown-link">

<el-icon class="el-icon--right"><More/>

</el-icon>

</span>

<template #dropdown>

<el-dropdown-item @click="setDropdownCardInfo('start', item)">起点</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('360', item)">图片</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('guidepost', item)">路标</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('hint', item)">提示</el-dropdown-item>

</template>

</el-dropdown>

<img :src="getUrl(item.url)" alt="" class="preview-image">

<div class="card-bottom">

<span class="tip">{{ this.groupingTips(item) }}</span>

<el-checkbox v-model="item.selected" class="checkbox"></el-checkbox>

</div>

</div>

</el-card>

<el-card v-for="(item, index) in dropdownCardInfo.hint"

:key="item.uid"

class="box-card">

<div class="image-container">

<el-dropdown trigger="click" class="el-dropdown">

<span class="el-dropdown-link">

<el-icon class="el-icon--right"><More/>

</el-icon>

</span>

<template #dropdown>

<el-dropdown-item @click="setDropdownCardInfo('start', item)">起点</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('360', item)">图片</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('guidepost', item)">路标</el-dropdown-item>

<el-dropdown-item @click="setDropdownCardInfo('hint', item)">提示</el-dropdown-item>

</template>

</el-dropdown>

<img :src="getUrl(item.url)" alt="" class="preview-image">

<div class="card-bottom">

<span class="tip">{{ this.groupingTips(item) }}</span>

<el-checkbox v-model="item.selected" class="checkbox"></el-checkbox>

</div>

</div>

</el-card>

</el-row>

</div>

</div>

<el-row>

<div class="info">

<span class="info-item" :class="{'red-text': dropdownCardInfo['start'] === null}">{{

'起点：' + (dropdownCardInfo['start'] ? 1 : 0)

}}</span>

<span class="info-item">{{ '图片：' + dropdownCardInfo['360'].length }}</span>

<span class="info-item">{{ '路标：' + dropdownCardInfo['guidepost'].length }}</span>

<span class="info-item">{{ '提示：' + dropdownCardInfo['hint'].length }}</span>

</div>

</el-row>

<template #footer>

<span class="dialog-footer">

<el-button @click="handleClose">取消</el-button>

<el-button type="primary" @click="handConfirm">

确定

</el-button>

</span>

</template>

</el-dialog>

</template>

<style scoped>

.el-upload-button {

width: 100px;

height: 40px;

margin-left: 8px;

}

.upload-demo {

display: inline;

}

.card-row {

display: flex;

flex-wrap: wrap;

justify-content: flex-start;

}

.box-card {

flex: 0 0 calc(33.3% - 10px);

margin: 5px;

position: relative;

box-sizing: border-box;

border: 1px solid #ddd;

border-radius: 4px;

}

.preview-image {

width: 100%;

height: 150px;

}

.image-container {

position: relative;

width: calc(100% - 20px);

height: calc(100% - 35px);

}

.scroll-container {

max-height: 454px;

overflow-y: auto;

}

.el-dropdown {

float: right;

}

.scroll-container::-webkit-scrollbar {

width: 8px;

}

.red-text {

color: red

}

.scroll-container::-webkit-scrollbar-thumb {

background-color: #8c939d;

border-radius: 4px;

}

.scroll-container::-webkit-scrollbar-track {

background-color: #f0f0f0;

}

.scroll-container::-webkit-scrollbar-button {

display: none; /\* 隐藏按钮 \*/

}

.checkbox {

float: right;

z-index: 1;

zoom: 150%;

}

.info {

font-size: 16px;

font-weight: 500;

color: #222226;

display: flex;

transform: translateY(12px);

}

.card-bottom {

display: flex;

justify-content: space-between;

align-items: center;

}

.info-item {

margin-right: 20px;

}

</style><template>

<div>

<canvas ref="canvasRef" class="three" id="three"></canvas>

<SelectDialog

v-model="selectDialogVisible"

:selectitems="selectitems"

:tip="tip"

@selectedCard="handleSelectedCard"

></SelectDialog>

<TextDialog

v-model="textDialogVisible"

:tip="tip"

@handText="handText"

></TextDialog>

</div>

<div class="content">

<div class="tooltip" id="tooltip">{{ tooltopContent }}</div>

</div>

</template>

<script setup>

import \* as THREE from 'three';

import {OrbitControls} from 'three/examples/jsm/controls/OrbitControls';

import {ref, defineProps, watch, defineEmits} from 'vue';

import SelectDialog from '@/components/panorama/SelectDialog.vue';

import TextDialog from '@/components/panorama/TextDialog.vue';

import {submit360} from '@/api/api';

import {useStore} from 'vuex';

import ContextMenu from "@imengyu/vue3-context-menu";

import {GUI} from 'three/addons/libs/lil-gui.module.min.js';

import {ElMessage} from "element-plus";

import router from "@/router";

const undetermined = ref(null); // 待定

const selectDialogVisible = ref(false);

const selectitems = ref(Array(null));

let isAlgin = false;

const textDialogVisible = ref(false);

const store = useStore();

const props = defineProps({

dropdownCardInfo: {

type: Object,

default: () => ({

start: null,

'360': [],

guidepost: [],

hint: [],

}),

},

isSend: Boolean,

describe: {

type: Object,

default: () => ({

address: [],

title: null,

}),

},

spots: Array,

});

const emit = defineEmits(['update-is-send']);

watch(() => props.isSend, (newVal) => {

if (newVal) {

var nid = store.getters.getUser.nid;

submit360(nid, dataList, props.describe).then(response => {

if (response.data.message == '1') {

ElMessage.success('上传成功')

router.push('/adminmanage')

} else if(response.data.message == '2'){

ElMessage.success('修改成功')

router.push('/adminmanage')

}else {

ElMessage.error(response.data.message)

}

})

}

emit('update-is-send', false);

});

watch(() => props.spots, (newVal) => {

dataList = newVal

for (var i = 0; i < dataList[0]['tips'].length; i++) {

changeAddSprite(dataList[0]['tips'][i])

}

})

const tip = ref(''); // 弹窗名称

const canvasRef = ref(null);

let scene;

let camera;

let sphere, controls, renderer, rayCaster;

let curr = 0;

let dataList = [];

}

function switchScene(url, index) {

canvasRef.value.removeEventListener('contextmenu', oncontextmenu);

canvasRef.value.removeEventListener('click', onClick);

canvasRef.value.removeEventListener('mousemove', onMousemove, false);

if (url.url) {

currImage = url;

} else {

currImage['url'] = url

}

curr = index

scene = null;

sphere = null;

initThree()

if (currImage.url.startsWith("blob:")) {

loadModel(currImage.url);

} else {

loadModel('/api' + currImage.url);

}

for (let i = 0; i < dataList[curr].tips.length; i++) {

changeAddSprite(dataList[curr]['tips'][i])

}

canvasRef.value.addEventListener('contextmenu', oncontextmenu);

canvasRef.value.addEventListener('click', onClick);

canvasRef.value.addEventListener('mousemove', onMousemove, false);

}

function onClick(event) {

event.preventDefault();

let x = ((event.clientX - document.getElementById("three").getBoundingClientRect().left) / document.getElementById("three").offsetWidth) \* 2 - 1;

let y = -((event.clientY - document.getElementById("three").getBoundingClientRect().top) / document.getElementById("three").offsetHeight) \* 2 + 1;

let standardVector = new THREE.Vector3(x, y, 1);// 标准设备坐标

let worldVector = standardVector.unproject(camera);

let ray = worldVector.sub(camera.position).normalize();

rayCaster.set(camera.position, ray);

const objs = rayCaster.intersectObjects(sphere.children, true);

if (objs.length > 0) {

const spriteClick = objs[0].object;

let index = sphere.children.findIndex((sprite) => sprite === spriteClick);

onGui(spriteClick, index)

}

}

function oncontextmenu(e) {

e.preventDefault()

ContextMenu.showContextMenu({

x: e.x,

y: e.y,

items: [

{

label: '添加路标',

onClick: async () => {

selectitems.value = props.dropdownCardInfo['guidepost'];

tip.value = '路标'

selectDialogVisible.value = true;

if (selectDialogVisible.value) {

await waitSelectDialogClose();

if (undetermined !== null) {

let mouse = new THREE.Vector2(

(e.clientX / window.innerWidth) \* 2 - 1,

-(e.clientY / window.innerHeight) \* 2 + 1

);

rayCaster.setFromCamera(mouse, camera);

let intersects = rayCaster.intersectObject(sphere);

if (intersects.length > 0) {

let point = intersects[0].point;

if (undetermined.value != null) {

let tempData = filterSelect(undetermined.value,)

addSprite(point, tempData, '路标');

undetermined.value = null

}

}

}

}

},

},

{

label: '添加提示',

onClick: async () => {

selectitems.value = props.dropdownCardInfo['hint'];

tip.value = '提示'

selectDialogVisible.value = true;

if (selectDialogVisible.value) {

await waitSelectDialogClose();

if (undetermined !== null) {

let mouse = new THREE.Vector2(

(e.clientX / window.innerWidth) \* 2 - 1,

-(e.clientY / window.innerHeight) \* 2 + 1

);

rayCaster.setFromCamera(mouse, camera);

let intersects = rayCaster.intersectObject(sphere);

if (intersects.length > 0) {

let point = intersects[0].point;

if (undetermined.value != null) {

let tempData = filterSelect(undetermined.value)

addSprite(point, tempData, '提示');

undetermined.value = null

}

}

}

}

}

}

]

})

}

function changeAddSprite(data) {

let spriteMap;

if (data.image.url.startsWith("blob:")) {

spriteMap = new THREE.TextureLoader().load(data.image.url);

} else {

spriteMap = new THREE.TextureLoader().load('/api' + data.image.url);

}

spriteMap.needsUpdate = true

let spriteMaterial = new THREE.SpriteMaterial({

map: spriteMap,

transparent: true,

alphaTest: 0.5,

});

let newsprite = new THREE.Sprite(spriteMaterial);

newsprite.name = data.tip

newsprite.position.copy(data.position);

newsprite.scale.set(5, 5, 1)

newsprite.center = new THREE.Vector2(0.5, 0)

newsprite.content = data.content;

sphere.add(newsprite);

}

function addSprite(point, info, tip) {

try {

let spriteMap

if (info.url.startsWith("blob:")) {

spriteMap = new THREE.TextureLoader().load(info.url);

} else {

spriteMap = new THREE.TextureLoader().load('/api' + info.url);

}

spriteMap.needsUpdate = true

let spriteMaterial = new THREE.SpriteMaterial({

map: spriteMap,

transparent: true,

alphaTest: 0.5,

});

let newsprite = new THREE.Sprite(spriteMaterial);

newsprite.name = tip

newsprite.position.copy(point);

newsprite.scale.set(5, 5, 1)

newsprite.center = new THREE.Vector2(0.5, 0)

const labelData = {

image: info, // 同样根据需求设置图片链接

tip: tip,

scale: {

x: newsprite.scale.x,

y: newsprite.scale.y,

z: newsprite.scale.z

},

position: {

x: newsprite.position.x,

y: newsprite.position.y,

z: newsprite.position.z,

},

content: {

text: null, // 设置标签文本

next: -1, // 没有连接的图片

showTip: false, // 是否显示弹窗

},

}

newsprite.content = labelData.content;

sphere.add(newsprite);

dataList[curr].tips.push(labelData)

} catch (error) {

alert('未选择图片资源')

}

}

async function waitSelectDialogClose() {

return new Promise((resolve) => {

const interval = setInterval(() => {

if (!selectDialogVisible.value) {

clearInterval(interval);

resolve();

}

}, 100);

});

}

let gui = null

function filterSelect(item) {

for (let arrayType in props.dropdownCardInfo) {

if (props.dropdownCardInfo[arrayType] !== null && arrayType !== 'start') {

for (let i = 0; i < props.dropdownCardInfo[arrayType].length; i++) {

if (props.dropdownCardInfo[arrayType][i].uid == item.uid) {

return props.dropdownCardInfo[arrayType][i];

}

}

} else if (arrayType == 'start' && props.dropdownCardInfo[arrayType] !== null) {

if (props.dropdownCardInfo[arrayType].uid === item.uid) {

return props.dropdownCardInfo[arrayType]

}

}

}

}

function onGui(sprite, Index) {

try {

if (gui) {

gui.destroy()

}

gui = new GUI();

gui.add(sprite.position, 'x', -100, 180).name('位置-x').onChange(function (value) {

dataList[curr]['tips'][Index].position.x = value

});

gui.add(sprite.position, 'y', -100, 180).name('位置-y').onChange(function (value) {

dataList[curr]['tips'][Index].position.y = value;

});

gui.add(sprite.position, 'z', -100, 180).name('位置-z').onChange(function (value) {

dataList[curr]['tips'][Index].position.z = value;

});

gui.add(sprite.scale, 'x', 0, 20).name('大小-x').onChange(function (value) {

dataList[curr]['tips'][Index].scale.x = value;

});

gui.add(sprite.scale, 'y', 0, 20).name('大小-y').onChange(function (value) {

dataList[curr]['tips'][Index].scale.y = value;

});

gui.add({

'校准': function () {

camera.lookAt(sprite.position);

}

}, '校准');

gui.add({

'提示': function () {

const newText = prompt("请输入提示文字：");

if (newText) {

dataList[curr]['tips'][Index]['content']['text'] = newText;

dataList[curr]['tips'][Index]['content']['showTip'] = true;

sprite.content.text = newText;

sprite.content.showTip = true;

onGui(sprite, Index)

}

}

}, '提示');

if (sprite.content.showTip) {

gui.add({

'清空文字': function () {

dataList[curr]['tips'][Index]['content']['text'] = null;

dataList[curr]['tips'][Index]['content']['showTip'] = false;

sprite.content.text = null;

sprite.content.showTip = false;

onGui(sprite, Index)

}

}, '清空文字')

}

gui.add({

'链接': async function () {

selectitems.value = props.dropdownCardInfo['360'].concat(props.dropdownCardInfo["start"]);

tip.value = '提示'

selectDialogVisible.value = true;

if (selectDialogVisible.value) {

await waitSelectDialogClose();

if (undetermined.value !== null) {

let idx = -1;

dataList.forEach(function (data, index) {

if (data.image.url == undetermined.value.url) {

idx = index

}

})

if (idx === -1) idx = dataList.length

let dict = {

image: filterSelect(undetermined.value),

tips: []

}

dataList.push(dict)

dataList[curr]['tips'][Index]['content']['next'] = idx;

sprite.content.next = idx;

undetermined.value = null

onGui(sprite, Index);

}

}

}

}, '链接');

if (dataList[curr]['tips'][Index].content.next !== -1) {

gui.add({

'切换场景': function () {

canvasRef.value.removeEventListener('contextmenu', oncontextmenu)

canvasRef.value.removeEventListener('click', onClick)

canvasRef.value.removeEventListener('mousemove', onMousemove, false)

sphere = null;

let index = dataList[curr]['tips'][Index].content.next;

switchScene(dataList[index].image.url, index)

}

}, '切换场景')

}

gui.add({

'删除': function () {

if (sprite) {

if (dataList[curr]['tips'][Index].content.next !== -1) {

let cnt = -1;

dataList.forEach((function (data, index) {

data.tips.forEach(function (item) {

if (item.content.next == dataList[curr]['tips'][Index].content.next) cnt++;

if (cnt > 0) return null;

})

if (cnt > 0) return null;

}))

if (cnt > 0) {

dataList[curr].tips.splice(Index, 1);

sprite.parent.remove(sprite);

} else {

dataList.forEach(function (data) {

data.tips.forEach(function (item) {

if (item.content.next > dataList[curr]['tips'][Index].content.next)

item.content.next--;

})

})

let deleIndex = dataList[curr]['tips'][Index].content.next;

dataList[curr].tips.splice(Index, 1);

dataList.splice(deleIndex, 1)

sprite.parent.remove(sprite);

}

}

sprite = null; // 将 sprite 置为 null，表示已被删除

}

gui.hidden = false

gui.destroy()

}

}, '删除');

gui.add({

'确定': function () {

gui.hidden = true

undetermined.value = selectedCard;

};

const handText = (msg) => {

undetermined.value = msg;

};

function onMousemove(e) {

let three = document.getElementById("three")

let x = ((e.clientX - three.getBoundingClientRect().left) / three.offsetWidth) \* 2 - 1;

let y = -((e.clientY - three.getBoundingClientRect().top) / three.offsetHeight) \* 2 + 1;

let mouse = new THREE.Vector2(x, y);

rayCaster.setFromCamera(mouse, camera)

let intersects = rayCaster.intersectObjects(sphere.children, true);

let tooltip = document.getElementById('tooltip');

if (intersects.length > 0) {

let p = intersects[0].object.position.clone().project(camera);

if (intersects[0].object.content.showTip) {

tooltopContent.value = intersects[0].object.content.text;

tooltip.style.top = ((-1 \* p.y + 1) \* three.offsetHeight / 2) + 'px';

tooltip.style.left = ((p.x + 1) \* three.offsetWidth / 2) + 'px'

}

} else {

tooltip.style.top = '0';

tooltip.style.left = '0'

tooltopContent.value = null

}

}

</script>

<style scoped>

.three {

width: 100%;

height: 100vh;

display: block;

}

.content {

font-family: serif;

color: rgba(0, 0, 0, .77);

}

.tooltip {

position: absolute;

top: 0;

left: 0;

padding: 10px;

transform: translate3d(-70%, -110%, 0);

transition: opacity .3s;

background-color: rgba(0, 0, 0, 0.8); /\* 使用rgba来设置黑色背景和透明度 \*/

box-shadow: 0 0 10px rgba(0, 0, 0, 0.5); /\* 添加阴影效果 \*/

color: #fff; /\* 设置文本颜色为白色，以确保文本在黑色背景上可见 \*/

}

</style><script>

import {ref} from "vue";

import AddImageDialog from "@/components/panorama/AddImageDialog.vue";

import EffectDisplay from "@/components/panorama/EffectDisplay.vue";

import PcaArr from "@/components/panorama/PcaArr.vue";

import TextDialog from "@/components/panorama/TextDialog.vue";

import {get360} from "@/api/api";

import {useRoute} from "vue-router";

export default {

components: {

TextDialog,

EffectDisplay,

AddImageDialog,

PcaArr

},

data() {

return {

spots: ref(null),

dialogTableVisible: ref(false),

dropdownCardInfo: ref({

'start': null,

'360': [],

'guidepost': [],

'hint': []

}),

isSend: ref(false),

pcaDialogVisible: ref(false),

describe: ref({

'address': [],

'title': null

}),

textDialogVisible: ref(false)

}

},

methods: {

updateDropdownCardInfo(newVal) {

this.dropdownCardInfo = newVal

},

updateIsSend(newVal) {

this.isSend = newVal

},

updateAddress(newVal) {

this.describe['address'] = newVal.split(' ')

},

updateTitle(newVal) {

this.describe['title'] = newVal

}

},

created() {

const route = useRoute();

const id = route.query.id

if (id) {

get360(id).then(responser => {

const data = responser.data.data;

this.spots = data['spots']

data['spots'].forEach((item, index) => {

if (index == 0) {

this.dropdownCardInfo.start = item.image;

} else {

this.dropdownCardInfo["360"].push(item.image);

}

item.tips.forEach((tip) => {

if (tip.tip == '路标') {

this.dropdownCardInfo['guidepost'].push(tip.image)

} else {

this.dropdownCardInfo['hint'].push(tip.image)

}

})

})

this.describe['address'] = [data['province'], data['city'], data['county'], data['address']]

this.describe['title'] = data['title']

this.describe['id'] = id;

})

}

}

}

</script>

<template>

<el-container class="container">

<el-aside width="100px">

<el-menu

class="el-menu-vertical-demo"

>

<el-menu-item @click="dialogTableVisible = true">添加图片</el-menu-item>

<el-menu-item @click="pcaDialogVisible = true">添加地址</el-menu-item>

<el-menu-item @click="textDialogVisible = true">标题</el-menu-item>

<el-menu-item @click="isSend = true">完成</el-menu-item>

</el-menu>

</el-aside>

<el-container>

<el-main>

<AddImageDialog v-show="dialogTableVisible"

v-model="dialogTableVisible"

@update:dialog-table-visible="dialogTableVisible = $event"

@update:dropdown-card-info="updateDropdownCardInfo"

:info="dropdownCardInfo"></AddImageDialog>

<EffectDisplay :dropdownCardInfo="dropdownCardInfo"

:isSend="isSend"

@update-is-send="updateIsSend"

:describe="describe"

:spots="spots"></EffectDisplay>

<PcaArr v-if="pcaDialogVisible"

v-model="pcaDialogVisible"

@update-is-address="updateAddress"

:info="describe['address']"></PcaArr>

<TextDialog v-if="textDialogVisible"

v-model="textDialogVisible"

@handText="updateTitle"

:info="describe['title']"></TextDialog>

</el-main>

</el-container>

</el-container>

</template>

<style scoped>

.container {

height: 100vh;

display: flex;

}

.el-menu-vertical-demo {

flex: none;

}

</style><script setup>

import {onMounted, ref} from "vue";

import {pcaTextArr} from 'element-china-area-data'

const props = defineProps({

pcaDialogVisible: Boolean,

info: Array

})

const emit = defineEmits(['update:modelValue', 'update-is-address'])

const selectedOptions = ref(props.info ? [props.info[0], props.info[1], props.info[2]] : []);

const address = ref(props.info ? props.info[3] : null);

const handleClose = () => {

emit('update:modelValue', false)

}

const handConfirm = () => {

emit('update-is-address', selectedOptions.value.join(' ') + ' ' + address.value)

emit('update:modelValue', false)

}

</script>

<template>

<el-dialog

v-model="props.pcaDialogVisible"

title="地址"

width="32%"

:before-close="handleClose"

>

<el-row>

<el-col :span="12">

<el-cascader

size="large"

:options="pcaTextArr"

v-model="selectedOptions">

</el-cascader>

</el-col>

<el-col :span="12">

<el-input

class="text"

v-model="address"

autosize

placeholder="详细地址"

/>

</el-col>

</el-row>

<template #footer>

<span class="dialog-footer">

<el-button @click="handleClose">取消</el-button>

<el-button type="primary" @click="handConfirm">

确定

</el-button>

return "超级管理员";

} else if (this.formLabelAlign.is\_admin) {

return "管理员";

} else if (!this.formLabelAlign.is\_superuser && !this.formLabelAlign.is\_admin) {

return "用户";

}

},

handleConfirm() {

fetchBlobData(this.formLabelAlign.avatar).then(async blobData => {

if (blobData instanceof Blob) {

blobData = await blobToBase64(blobData);

}

const userData = {

nid: this.store.getters.getUser.nid,

avatar: blobData,

email: this.formLabelAlign.email,

username: this.formLabelAlign.username,

is\_new\_pasword: this.formLabelAlign.is\_new\_pasword, // 注意这里的初始值是布尔类型

password: this.formLabelAlign.pasword,

new\_password: this.formLabelAlign.new\_password

}

changeUserInfo(userData).then(response => {

if (response.data.message == '修改成功') {

ElMessage.success('修改成功');

} else {

ElMessage.error(response.data.message)

}

})

})

},

handleAvatarSuccess(response, uploadFile) {

this.formLabelAlign.avatar = URL.createObjectURL(uploadFile[0].raw)

}

}

};

</script>

<template>

<div class="form-container">

<el-form

ref="ruleFormRef"

label-position="left"

label-width="100px"

:model="formLabelAlign"

:rules="ruleForm"

style="max-width: 460px"

>

<el-form-item label="头像">

<el-upload

class="avatar-uploader"

:show-file-list="false"

:on-change="handleAvatarSuccess"

:auto-upload="false"

>

<img v-if="formLabelAlign['avatar']" :src="formLabelAlign['avatar']" class="avatar" alt=""/>

<el-icon v-else class="avatar-uploader-icon">

<Plus/>

</el-icon>

</el-upload>

</el-form-item>

<el-form-item label="用户名">

<el-input v-model="formLabelAlign.username"/>

</el-form-item>

<el-form-item label="邮箱">

<el-input v-model="formLabelAlign.email"/>

</el-form-item>

<el-form-item label="设置新密码">

<el-switch

v-model="formLabelAlign.is\_new\_pasword"

class="mb-2"

/>

</el-form-item>

<el-form-item label="原密码" v-if="formLabelAlign.is\_new\_pasword" prop="password">

<el-input v-model="formLabelAlign.pasword"/>

</el-form-item>

<el-form-item label="新密码" v-if="formLabelAlign.is\_new\_pasword" prop="new\_password">

<el-input v-model="formLabelAlign.new\_password"/>

</el-form-item>

<el-form-item>

<el-button type="primary" @click="handleConfirm">

确定

</el-button>

</el-form-item>

</el-form>

</div>

</template>

<style scoped>

.form-container {

margin: 120px auto;

max-width: 600px; /\* 调整适当的最大宽度 \*/

}

.avatar-uploader .avatar {

width: 120px;

height: 120px;

display: block;

}

</style>

<style>

.avatar-uploader .el-upload {

border: 1px dashed var(--el-border-color);

border-radius: 6px;

cursor: pointer;

position: relative;

overflow: hidden;

transition: var(--el-transition-duration-fast);

}

.avatar-uploader .el-upload:hover {

border-color: var(--el-color-primary);

}

.el-icon.avatar-uploader-icon {

font-size: 28px;

color: #8c939d;

width: 120px;

height: 120px;

text-align: center;

}

</style><script>

import PersonalInformation from "@/components/people/PersonalInformation.vue";

import ChangeInfo from "@/components/people/ChangeInfo.vue";

export default {

components: {

PersonalInformation,

ChangeInfo

},

data() {

return {

menuActive: 1

}

},

methods: {

handleMenuSelect(index) {

this.menuActive = index

}

}

}

</script>

<template>

<el-row class="tac">

<el-col :span="3">

<el-menu

class="el-menu-vertical-demo"

style="height: 100vh"

@select="handleMenuSelect"

default-active="1"

v-model="menuActive"

>

<el-menu-item index="1">基本信息</el-menu-item>

<el-menu-item index="2">修改资料</el-menu-item>

</el-menu>

</el-col>

<el-col :span="21">

<PersonalInformation v-if="menuActive == 1"/>

<ChangeInfo v-if="menuActive == 2"/>

</el-col>

</el-row>

</template>

<style scoped>

</style><script>

import {onMounted, ref} from "vue";

import {useStore} from "vuex";

import {getInfo} from "@/api/api";

import {ElMessage} from "element-plus";

import {formatTimestamp} from "@/utils/BlobData";

export default {

data() {

return {};

},

setup() {

const formLabelAlign = ref({

avatar: '',

create\_time: '',

email: '',

is\_admin: '',

is\_superuser: '',

username: ''

});

const store = useStore();

onMounted(() => {

let nid = store.getters.getUser.nid;

getInfo(nid).then(response => {

if (response.data.msg == 'succeed') {

formLabelAlign.value = response.data.data

console.log(formLabelAlign.value)

} else {

ElMessage.error('获取信息失败')

}

}).catch(error => {

ElMessage.error('获取信息失败')

console.log(error)

})

})

return {

formLabelAlign

}

},

methods: {

formatTimestamp,

getGroup() {

if (this.formLabelAlign.is\_superuser) {

return "超级管理员";

} else if (this.formLabelAlign.is\_admin) {

return "管理员";

} else if (!this.formLabelAlign.is\_superuser && !this.formLabelAlign.is\_admin) {

return "用户";

}

}

}

};

</script>

<template>

<div class="form-container">

<el-form

label-position="left"

label-width="100px"

:model="formLabelAlign"

size="large"

>

<el-form-item label="头像:">

<el-avatar :src="'api' + formLabelAlign.avatar" :size="120"/>

</el-form-item>

<el-form-item label="用户名:">

<el-text>{{ formLabelAlign.username }}</el-text>

</el-form-item>

<el-form-item label="邮箱:">

<el-text>{{ formLabelAlign.email }}</el-text>

</el-form-item>

<el-form-item label="所属用户组:">

<el-text>{{ getGroup() }}</el-text>

</el-form-item>

<el-form-item label="注册时间:">

<el-text>{{ formatTimestamp(formLabelAlign.create\_time) }}</el-text>

</el-form-item>

</el-form>

</div>

</template>

<style scoped>

.form-container {

margin: 120px auto;

max-width: 600px; /\* 调整适当的最大宽度 \*/

}

.el-avatar {

width: 120px; /\* 调整适当的头像大小 \*/

height: 120px;

}

.form {

margin: auto auto;

}

</style>import {createRouter, createWebHistory} from "vue-router"

const routes = [

{//3D展馆

path: '/',

name: 'Home',

component: () => import(/\* webpackChunkName: "Home" \*/"@/views/Three360.vue")

},

{//3D展示

path: '/display/:username',

name: 'display',

component: () => import(/\* webpackChunkName: "Home" \*/"@/components/3Ddisplay.vue")

},

{

path: '/login',

name: 'Login',

component:()=>import(/\* webpackChunkName: "Home" \*/"@/views/Login.vue")

},

{

path: '/register',

name: 'Register',

component:()=>import(/\* webpackChunkName: "Home" \*/"@/views/Register.vue")

},

{

path: '/adminmanage',

name: 'AdminManage',

component: () => import(/\* webpackChunkName: "Home" \*/"@/views/AdminManage.vue")

},

{

path: '/people',

name: 'PersonalInformation',

component: () => import(/\* webpackChunkName: "Home" \*/"@/components/people/PeopleMenu.vue")

}

]

const router = createRouter({

history: createWebHistory(),

routes,

})

export default routerimport { createStore } from 'vuex'

import user from "@/store/modules/user";

export default createStore({

modules:{

user

}

});

import jwtDecode from "jwt-decode";

import axiosInstance from "@/api";

const axios = axiosInstance

const user = {

state: {

user: {

username: '',

avatar: '/media/logout.png',

nid: '',

access: '',

refresh: '',

is\_login: false

},

id: null

},

getters: {

getUser(state) {

return state.user

},

getId(state) {

return state.id

}

},

mutations: {

updateUser(state, user) {

state.user.username = user.username

state.user.avatar = user.avatar

state.user.nid = user.nid

state.user.access = user.token

state.user.refresh = user.refresh

state.user.is\_login = user.is\_login

},

updataId(state, id) {

state.id = id

},

updateAccess(state, user) {

state.user.access = user.access

},

logout(state) {

state.user.username = '';

state.user.avatar = '/media/logout.png';

state.user.nid = '';

state.user.access = '';

state.user.refresh = '';

state.user.is\_login = false;

}

},

actions: {

asyncUpdateUser(context, user) {

context.commit('updateUser', user);

},

login: (context, data) => {

axios.post('/api/login/', {

user: data.user,

pwd: data.pwd,

valid\_code: data.valid\_code

}).then(response => {

const {access, refresh} = response.data

const access\_obj = jwtDecode(access)

setInterval(() => {

axios.post('/api/refresh/', {

refresh

}).then(response => {

context.commit("updateAccess", response.data.access)

}).catch(error => {

console.log(error)

})

}, 4.5 \* 60 \* 1000);

axios.get('/api/userinfo/', {

params: {

nid: access\_obj.nid

},

headers: {

'Authorization': `Bearer ${access}`

}

}).then(response => {

context.commit("updateUser", {

...response.data,

nid: access\_obj.nid,

access: access,

refresh: refresh,

is\_login: true

});

data.success();

}).catch(error => {

console.error(error);

data.error();

})

})

}

}

}

export default userexport const fetchBlobData = async (blobUrl) => {

const response = await fetch(blobUrl);

return await response.blob();

}

export const blobToBase64 = (blob) => {

return new Promise((resolve, reject) => {

const reader = new FileReader();

reader.onload = () => resolve(reader.result);

reader.onerror = reject;

reader.readAsDataURL(blob);

});

};

export const formatTimestamp = (timestampStr) => {

const timestamp = new Date(timestampStr);

const year = timestamp.getFullYear();

const month = (timestamp.getMonth() + 1).toString().padStart(2, '0');

const day = timestamp.getDate().toString().padStart(2, '0');

const hours = timestamp.getHours().toString().padStart(2, '0');

const minutes = timestamp.getMinutes().toString().padStart(2, '0');

return `${year}-${month}-${day} ${hours}:${minutes}`;

}import \* as THREE from 'three'

<script>

import {Document, Location, PictureFilled, Setting, User} from "@element-plus/icons-vue";

import UserManagement from "@/components/AdminManage/UserManagement.vue";

import PanoramaManagement from "@/components/AdminManage/PanoramaManagement.vue";

export default {

components: {PanoramaManagement, UserManagement, PictureFilled, User, Setting, Document, Location},

data() {

return {

selectedMenu: localStorage.getItem("selectedMenu") || "UserManagement",

}

},

methods: {

handleMenuSelect(index) {

this.selectedMenu = index

localStorage.setItem("selectedMenu", index);

}

}

}

</script>

<template>

<el-row>

<el-col :span="3">

<el-menu

class="el-menu-vertical-demo"

@select="handleMenuSelect"

default-active="UserManagement"

>

<el-menu-item index="UserManagement">

<el-icon>

<User/>

</el-icon>

<span>用户管理</span>

</el-menu-item>

<el-menu-item index="PanoramaManagement">

<el-icon>

<PictureFilled/>

</el-icon>

<span>全景管理</span>

</el-menu-item>

</el-menu>

</el-col>

<el-col :span="21">

<div class="content-container">

<div v-if="selectedMenu === 'UserManagement'">

<UserManagement/>

</div>

<div v-else-if="selectedMenu === 'ArticleManagement'">

<ArticleManage/>

</div>

<div v-else-if="selectedMenu === 'PanoramaManagement'">

<PanoramaManagement/>

</div>

</div>

</el-col>

</el-row>

</template>

<style scoped>

.el-menu-vertical-demo {

height: 100vh;

}

.content-container {

flex-direction: row;

flex-wrap: nowrap;

}

</style><template>

<MainMenu></MainMenu>

<div class="registerForm">

<el-form

label-position="top"

ref="ruleFormRef"

:model="ruleForm"

:rules="rules"

hide-required-asterisk>

<el-form-item label="用户名" prop="username">

<el-input v-model="username"></el-input>

</el-form-item>

<el-form-item label="请输入密码" prop="password">

<el-input v-model="password"></el-input>

</el-form-item>

<el-form-item label="验证码" prop="verifyCode">

<el-row>

<el-col :span="16">

<el-input v-model="valid\_code"></el-input>

</el-col>

<el-col :span="8">

<el-image :src="imageSrc" @click="loadVerCodeImg"/>

</el-col>

</el-row>

</el-form-item>

<div class="resBtn">

<el-button type="primary" @click="login">登录</el-button>

</div>

</el-form>

</div>

</template>

<script>

import {getVercodeimg, loginRequest} from "@/api/api";

import MainMenu from "@/components/MainMenu.vue";

import Store from "@/store";

import {ref, onMounted} from "vue";

import router from "@/router";

import {ElMessage} from "element-plus";

export default {

components: {

MainMenu,

},

setup() {

const imageSrc = ref("");

const username = ref("");

const password = ref("");

const error\_message = ref("");

const valid\_code = ref("");

const loadVerCodeImg = () => {

getVercodeimg()

.then((response) => {

const data = new Uint8Array(response.data);

const base64 = "data:image/png;base64," + btoa(String.fromCharCode(...data));

imageSrc.value = base64;

})

.catch((error) => {

ElMessage.error("获取验证码失败")

});

};

const validateVerCode = (rule, value, callback) => {

if (error\_message.value === "验证码错误") {

loadVerCodeImg();

callback(new Error(error\_message.value));

} else if (error\_message.value === null) {

callback();

}

};

const validateUser = (rule, value, callback) => {

if (error\_message.value === "用户名或者密码错误") {

loadVerCodeImg();

callback(new Error(error\_message.value));

} else if (error\_message.value === null) {

callback();

}

};

const login = () => {

const store = Store;

store

.dispatch("login", {

user: username.value,

pwd: password.value,

valid\_code: valid\_code.value,

success() {

router.push("/");

},

error() {

error\_message.value = "用户名或密码错误";

},

})

.catch((error) => {

ElMessage.error(error)

});

};

onMounted(() => {

loadVerCodeImg();

});

return {

imageSrc,

username,

password,

error\_message,

valid\_code,

login,

loadVerCodeImg,

validateVerCode,

validateUser,

};

},

};

</script>

<style>

.registerForm{

width: 300px;

margin: 7% auto auto auto;

}

.el-input{

border: none;

flex: 1;

width: 200px;

height: 35px;

}

.el-image{

height: 35px;

width: 120px;

}

.resBtn{

margin: 0 auto;

}

.el-button{

width: 260px;

margin-right: 10px;

}

</style>

<template>

<MainMenu></MainMenu>

<div class="registerForm">

<el-form

label-position="top"

ref="ruleFormRef"

:model="ruleForm"

:rules="rules"

hide-required-asterisk>

<el-form-item label="用户名" prop="username">

<el-input v-model="ruleForm.username"></el-input>

</el-form-item>

<el-form-item label="密码" prop="password">

<el-input v-model="ruleForm.password"></el-input>

</el-form-item>

<el-form-item label="确认密码" prop="conpassword">

<el-input v-model="ruleForm.conpassword"></el-input>

</el-form-item>

<el-form-item label="邮箱" prop="emailprop">

<el-input v-model="ruleForm.emailprop"></el-input>

</el-form-item>

<el-form-item label="验证码" prop="verifyCode">

<el-row>

<el-col :span="16">

<el-input v-model="ruleForm.verifyCode"></el-input>

</el-col>

<el-col :span="8">

<el-button :disabled="sendTime > 0"

type="primary"

@click="sendBtn">{{ sendTime > 0 ? sendTime + 's' : '发送' }}

</el-button>

</el-col>

</el-row>

</el-form-item>

<div class="resBtn">

<el-button type="primary" @click="submitForm">注册</el-button>

</div>

</el-form>

</div>

</template>

<script>

import {getEmailCode, registerRequest} from "@/api/api";

import MainMenu from "@/components/MainMenu.vue";

export default {

components: {

MainMenu

},

data() {

return {

vercode: '',

imageSrc: '',

sendTime: 0,

timer: null,

msg: null,

ruleForm: {

username: '',

password: '',

conpassword: '',

verifyCode: '',

emailprop: ''

},

rules: {

username: [

{required: true, message: '用户名不能为空', trigger: 'blur'},

{validator: this.validateUserName.bind(this), trigger: 'blur'}

],

password: [

{required: true, message: '密码不能为空', trigger: 'blur'},

{min: 6, max: 20, message: '密码长度在6到20个字符之间', trigger: 'blur'}

],

conpassword: [

{required: true, message: '密码不能为空', trigger: 'blur'},

{validator: this.validateConPassword.bind(this), trigger: 'blur'}

],

verifyCode: [

{required: true, message: '验证码不能为空', trigger: 'submit'},

{validator: this.validateVerCode.bind(this), trigger: 'submit'}

],

emailprop: [

{required: true, message: '邮箱不能为空', trigger: 'blur'},

{validator: this.validateEmail.bind(this), trigger: 'blur'}

]

}

}

},

methods: {

submitForm() {

registerRequest(this.ruleForm.username, this.ruleForm.password, this.ruleForm.emailprop, this.ruleForm.verifyCode).then(

response => {

this.msg = response.data.msg

if (this.msg == '注册成功') {

this.$router.push('/')

}

}

)

},

validateConPassword(rule, value, callback) {

if (value !== this.ruleForm.password) {

callback(new Error('两次输入的密码不一致'));

} else {

callback();

}

},

validateVerCode(rule, value, callback) {

if (this.msg === '验证码错误') {

callback(new Error(this.msg))

} else if (this.msg === '验证码失效，请重新获取') {

this.getCode()

callback(new Error(this.msg))

}

},

validateEmail(rule, value, callback) {

const re = /^[a-zA-Z0-9][a-zA-Z0-9\_]+\@[a-zA-Z0-9]+\.[a-zA-Z]{2,5}(\.[a-zA-Z]{2,5})\*$/i;

if (value.match(re) !== null) {

callback()

} else if (this.msg === '邮箱已注册') {

callback(new Error(this.msg))

} else {

callback(new Error('邮箱格式错误'));

}

},

validateUserName(rule, value, callback) {

if (this.msg === '用户名已存在') {

callback(new Error(this.msg))

} else {

callback()

}

},

sendBtn() {

this.$refs.ruleFormRef.validateField(['username', 'password', 'conpassword', 'emailprop'], (valid) => {

if (valid) {

this.getCode();

}

});

},

getCode() {//获取邮箱验证码

getEmailCode(this.ruleForm.emailprop).then(response => {

if (this.$refs.ruleFormRef.validateField('emailprop')) {

if (response.data === '邮件发送成功') {

alert('发送成功')

} else {

alert("发送失败")

clearInterval(this.timer)

this.sendTime = 0

}

this.sendTime = 60;

this.timer = setInterval(() => {

this.sendTime--;

if (this.sendTime === 0) {

clearInterval(this.timer);

}

}, 1000)

}

}).catch(error => {

alert('请求错误')

})

}

},

created() {

},

beforeDestroy() {

clearInterval(this.timer);

}

}

</script>

<style>

.registerForm {

width: 300px;

margin: 0 auto;

top: 60px;

}

.el-input {

border: none;

flex: 1;

width: 200px;

height: 35px;

}

.el-image {

height: 35px;

width: 120px;

}

.resBtn {

margin: 0 auto;

}

.el-button {

width: 260px;

margin-right: 10px;

}

.el-col .el-button {

width: 100px;

}

</style>

<script>

import MainMenu from "@/components/MainMenu.vue";

import {onMounted, ref} from "vue";

import {get360List} from "@/api/api";

import router from "@/router";

import {ElMessage} from "element-plus";

export default {

components: {

MainMenu

},

data() {

return {

isImage: false

}

},

setup() {

const articleList = ref([])

const pageSize = ref(4);

const currpag = ref(1);

const cardList = ref([])

const load = (page = 1, pageSize = 4) => {

get360List(page, pageSize).then(response => {

articleList.value = response.data.data

cardList.value = paginatedCardRows()

}).catch(error => {

ElMessage.error(error)

})

}

const paginatedCardRows = () => {

const maxRows = 3; // 最大行数

const rowSize = 6; // 每行最多数量

const pages = [];

let currentPage = [];

for (let i = 0; i < articleList.value.length; i++) {

if (currentPage.length === rowSize) {

pages.push(currentPage);

currentPage = [];

}

currentPage.push(articleList.value[i]);

if (pages.length === maxRows) {

break;

}

}

if (currentPage.length > 0) {

pages.push(currentPage);

}

return pages;

};

onMounted(() => {

load()

})

const calculateColSpan = () => {

const screenWidth = window.innerWidth;

return screenWidth >= 1200 ? 4 : screenWidth >= 768 ? 6 : 12;

};

const calculateCardMaxHeight = () => {

const screenHeight = window.innerHeight;

return screenHeight >= 1080 ? '400px' : '300px'; // 根据屏幕高度调整

};

return {

cardList,

pageSize,

currpag,

articleList,

load,

paginatedCardRows,

calculateColSpan,

calculateCardMaxHeight

}

},

methods: {

pageClick(number) {

this.load(number, 18)

},

handDisplay(nid, title) {

this.$store.commit('updataId', nid);

router.push({

name: 'display',

params: {

username: title

}

})

}

}

}

</script>

<template>

<MainMenu></MainMenu>

<div class="header">

<span>3D展馆</span>

<div class="dot"></div>

</div>

<div class="card">

<el-row v-for="(row,rowIndex) in cardList"

:key="rowIndex"

class="card-row">

<el-col

v-for="item in row"

:key="item.nid"

class="card-col"

:span="calculateColSpan()">

<el-card :body-style="{padding:'10px'}"

shadow="never"

class="card-item" @click="handDisplay(item.nid,item.title)">

<img :src="'/api'+item.url"

class="image"

alt=""/>

<div style="padding: 14px">

<span v-html="item.title"></span>

<div class="bottom">

<time class="time">{{ item.update\_time }}</time>

</div>

</div>

</el-card>

</el-col>

</el-row>

</div>

<div class="paginationdiv">

<el-pagination background layout="prev, pager, next"

:total="cardList.length"

class="pagination"

:page-size="pageSize"

v-model:current-page="currpag"

@current-change="pageClick"/>

</div>

</template>