<template>

<div class="PasswordModify">

<!-- 面板 -->

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

<div slot="header" class="clearfix title">

<span class="el-icon-d-arrow-right"></span>

<span>修改密码</span>

</div>

<div class="text item">

<el-form

:model="passwordForm"

status-icon

:rules="rules"

ref="passwordForm"

label-width="100px"

class="demo-ruleForm"

size="small"

style="width:400px"

>

<!-- 旧密码 -->

<el-form-item label="旧密码" prop="oldPassword">

<el-input type="password" v-model="passwordForm.oldPassword" auto-complete="off"></el-input>

</el-form-item>

<!-- 密码 -->

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

<el-input type="password" v-model="passwordForm.newPassword" auto-complete="off"></el-input>

</el-form-item>

<!-- 确定密码 -->

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

<el-input type="password" v-model="passwordForm.checkPass" auto-complete="off"></el-input>

</el-form-item>

<!-- 按钮组 -->

<el-form-item>

<el-button type="danger" @click="submitForm('passwordForm')">提交</el-button>

<el-button @click="resetForm('passwordForm')">重置</el-button>

</el-form-item>

</el-form>

</div>

</el-card>

</div>

</template>

<script>

// 引入验证用户名函数

import { passwordReg} from "@/utils/validator/validator.js";

// 引入local

import local from '@/utils/local/local';

export default {

data(){

// 原密码验证

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

if(!value){

callback(new Error("密码不能为空"));//输出错误提示

}else{

// 发送请求到后端请求原密码

this.request.get('/staff/getoldpassword')

.then(res=>{

// 获取旧密码和输入的密码进行对比

if(value === res){

callback()

}else{

callback(new Error("密码错误"));//输出错误提示

}

})

.catch(err=>{

console.log(err);

})

}

}

// 密码验证

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

if(!value){

callback(new Error("密码不能为空"));//输出错误提示

}else if(!passwordReg(value)){

callback(new Error("密码不能含有非法字符，长度在4-10之间"));//输出错误提示

}else if(value===this.passwordForm.oldPassword){

callback(new Error("新密码不能和原密码一致"));//输出错误提示

}else{

if(this.passwordForm.checkPass){//如果确认密码不为空触发一致性验证

// 触发一致性验证

this.$refs.passwordForm.validateField("checkPass");

}

callback();//成功回调

}

}

// 确认密码验证

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

if(!value){

callback(new Error("确认密码不能为空"));//输出错误提示

}else if(value !== this.passwordForm.newPassword){

callback(new Error("两次输入不一致"));//输出错误提示

}else{

callback();//成功回调

}

}

return{

// 表单数据

passwordForm:{

oldPassword:"",

newPassword:"",

checkpass:""

},

// 验证规则

rules:{

// 旧密码验证

oldPassword:[

{required:true,validator: validateOldPassword , trigger: 'blur'}

],

// 密码验证

newPassword:[

{required:true,validator: validatePassword, trigger: 'blur' }

],

// 确认密码验证

checkPass:[

{required:true,validator: validateCheckPass, trigger: 'blur' }

]

}

}

},

methods:{

// 重置按钮事件

resetForm(formName) {

this.$refs[formName].resetFields();

},

// 提交按钮事件

submitForm(formName) {

this.$refs[formName].validate((valid) => {

// 如果成功

if (valid) {

// 提交数据给后端

let params = {

newPassword:this.passwordForm.newPassword

};

//将新密码发送给后端

this.request.post('/staff/modifypassword',params)

.then(res=>{

// 接收参数

let {code,msg}=res;

if(code===0){

// 成功提示

this.$message({

type:"success",

message:msg

})

// 清除存储

local.remove("qike\_2019\_system");

// 跳转路由

setTimeout(()=>{

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

},1000)

}else if(code === 1){

//失败提示

this.$message.error(msg);

}

})

.catch(err=>{

console.log(err);

})

} else {

return false;

}

});

},

}

}

</script>

<style lang="less">

@import "./PasswordModify.less";

</style>