**实验三 Vue组件与Ant Deign Vue实验**

**一、实验目的：**

（1）练习掌握组件注册

（2）练习掌握组件传参Props

（3）练习掌握自定义事件

（4）练习掌握模板引用

（5）练习掌握列表渲染

（6）练习掌握UI框架Ant Design Vue

**二、实验内容：**

（1）设计编写一个程序，用组件的方式实现todoList功能，该程序用于实现要完成的任务列表，可以新增任务、删除任务、设置已完成的任务和统计已完成任务数。设计界面如下：



代码：

TodoItem.vue:

<template>

*<!-- 单选框 事件名称 删除按钮 -->*

    <li>

        <input type="checkbox" v-model="todoitem.isDone" :style="taskDone">

        <span :style="taskDone">{{ props.todoitem.itemName }}</span>

        <input @click="deleteTask" type="button" value="删除">

    </li>

</template>

<script setup lang="ts">

import { defineProps,computed,reactive,defineEmits } from "vue"

*// 从父组件接收数据*

interface  TodoItem {

    itemName: string,

    isDone: boolean

}

const props = defineProps({

    todoitem: {

        type: Object as () => TodoItem,

        required: true

    }

})

*// 自动变更字体样式*

const todoitem = reactive(props.todoitem)

const taskDone = computed(()=>{

    if(todoitem.isDone){

        return {"text-decoration": "line-through"}

    }else{

        return {"text-decoration": "none"}

    }

})

*// 向父组件传递消息*

const emit = defineEmits(["deleteTask"])

const deleteTask = ()=>{

    emit("deleteTask",todoitem)

}

export interface TodoItemInterface {

    itemName: string,

    isDone: boolean

}

</script>

<style scoped>

</style>

运行截图：

Itemlist.vue

<template>

    <ul>

        <todoItem v-for="item in todoList" :todoitem="item" :key="item.itemName" @delete-task="deleteTask"></todoItem>

    </ul>

</template>

<script setup lang="ts">

import { defineProps, reactive, watch } from "vue"

import todoItem from "./todoItem.vue"

import Bus from "../bus/bus"

import { TodoItemInterface } from "./todoItem.vue"

let todoList: TodoItemInterface[] = reactive([

    {

        itemName: "吃饭",

        isDone: false

    },

    {

        itemName: "睡觉",

        isDone: false

    },

    {

        itemName: "打豆豆",

        isDone: false

    }

])

*// 接收子组件传递过来的消息*

const deleteTask = (todoitem: TodoItemInterface) => {

*// 数组中找到这个元素*

    let index = todoList.findIndex((item) => {

        return item.itemName === todoitem.itemName

    })

    todoList.splice(index, 1)

    console.log(todoList)

}

*// 接收bus传递过来的消息*

Bus.on('add-task', (todoitem: TodoItemInterface) => {

    todoList.push(todoitem)

})

Bus.on('deleteAllDoneTask', () => {

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

        let index = todoList.findIndex((item) => {

            return item.isDone == true

        })

        todoList.splice(index, 1)

    }

})

watch(todoList, (newVal, oldVal) => {

    console.log(newVal)

    console.log(oldVal)

    Bus.emit('doneNum', todoList.filter((item) => {

        return item.isDone

    }).length)

}, { deep: true, immediate: true })

</script>

<style scoped></style>

Header.vue

<template>

    代办事件:

    <input type="text" v-model="itemName">

    <button @click="addTask">add</button>

</template>

<script setup lang="ts">

import { defineProps,reactive,ref } from "vue"

import  {TodoItemInterface}  from "./todoItem.vue"

import Bus from "../bus/bus"

const itemName = ref('')

const addTask = ()=>{

    let todoitem:TodoItemInterface = {

        itemName: itemName.value,

        isDone: false

    }

    Bus.emit('add-task',todoitem)

    itemName.value = ''

}

</script>

<style scoped>

</style>

Footer.vue:

<template>

    <div>

        已完成:{{doneNumThis }}个

        <button @click="deleteAllDoneTask">删除所有已完成的任务</button>

    </div>

</template>

<script setup lang="ts">

import { defineProps, reactive, ref } from "vue"

import Bus from "../bus/bus"

let doneNumThis = ref(0)

*// 接收bus传递过来的消息*

Bus.on('doneNum', (doneNum:number) => {

    doneNumThis.value = doneNum

})

*// 删除所有已完成的任务*

const deleteAllDoneTask = () => {

    console.log('删除所有已完成的任务')

    Bus.emit('deleteAllDoneTask')

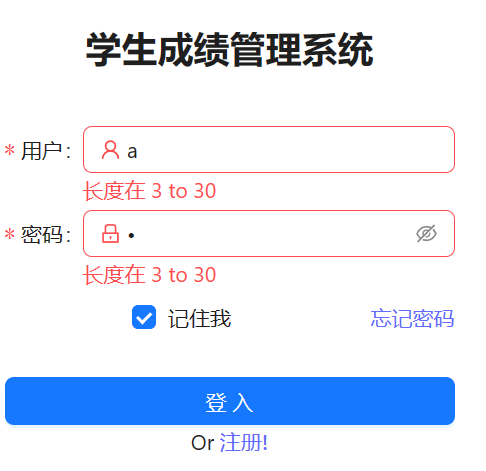
}

</script>

<style scoped></style>

运行截图：

1. 设计编写一个程序实现用户登录和用户注册的功能，要求用Ant Design Vue框架实现，在提交表单要进行必要的前端字段校验，只有校验通过才可提交表单。设计界面如下：



代码：

Login.vue:

// 用途: 登录页

<template>

    <el-row class="loginForm">

        <el-col>

            <el-form ref="loginForm" :model="ruleForm" status-icon :rules="rules" label-width="70px">

                <el-row justify="center" class="loginFormItem">

                    <el-col :xs="16" :sm="10" :md="8" :lg="6" :xl="6">

                        <el-form-item label="QQ 号 : " prop="qqNum">

                            <el-input v-model="ruleForm.qqNum" type="text" size="large" autocomplete="off" clearable />

                        </el-form-item>

                    </el-col>

                </el-row>

                <el-row justify="center" class="loginFormItem">

                    <el-col :xs="16" :sm="10" :md="8" :lg="6" :xl="6">

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

                            <el-input v-model="ruleForm.password" type="password" size="large" autocomplete="off"

                                show-password clearable />

                        </el-form-item>

                    </el-col>

                </el-row>

                <el-row justify="center" gutter="20" class="loginFormItem">

                    <el-col :xs="6" :sm="5" :md="4" :lg="2" :xl="2" align="middle">

                        <el-button type="primary" @click="submitForm(loginForm)" size="large">登录</el-button>

                    </el-col>

                    <el-col :xs="6" :sm="5" :md="4" :lg="2" :xl="2" align="middle">

                        <el-button @click="resetForm(loginForm)" size="large"

                            tag="a"

                            href="/register"

                            rel="noopener noreferrer"

                            style="text-decoration-line: *none*;"

                        >注册

                        </el-button>

                    </el-col>

                </el-row>

            </el-form>

        </el-col>

    </el-row>

</template>

<script lang="ts" setup>

import { reactive, ref } from 'vue'

import type { FormInstance, FormRules } from 'element-plus'

const loginForm = ref<FormInstance>()

const validatePass = (rule: any, value: any, callback: any) => {

    console.log(rule)

    console.log(value)

    console.log(callback)

    if (value === '') {

        callback(new Error('账号为空,请认真填写~'))

    }

}

const validatePass2 = (rule: any, value: any, callback: any) => {

    console.log(rule)

    if (value === '') {

        callback(new Error('密码为空,请认真填写~'))

    }

}

const ruleForm = reactive({

    qqNum: '',

    password: '',

})

const rules = reactive<FormRules<typeof ruleForm>>({

    qqNum: [{ validator: validatePass, trigger: 'blur' }],

    password: [{ validator: validatePass2, trigger: 'blur' }],

})

const submitForm = (formEl: FormInstance | undefined) => {

    if (!formEl) return

    formEl.validate((valid) => {

        if (valid) {

            console.log('submit!')

        } else {

            console.log('error submit!')

            return false

        }

    })

}

const resetForm = (formEl: FormInstance | undefined) => {

    if (!formEl) return

    formEl.resetFields()

}

</script>

<style scoped>

*/\**

*loginForm*

*表单在屏幕中央，*

*垂直居中，*

*随着屏幕的宽度而变化*

*\*/*

.loginForm {

    justify-content: *center*;

    align-items: *center*;

    height: 70vh;

}

.loginFormItem{

    height: 60px;

}

</style>

Register.vue:

// 用途: 登录页

<template>

    <el-row class="registerForm" justify="center">

        <el-col align="middle">

            <el-form ref="registerForm" :model="ruleForm" status-icon :rules="rules" label-width="120px"

                :label-position="labelPosition">

                <el-col :xs="inputItim.xs" :sm="inputItim.sm" :md="inputItim.md" :lg="inputItim.lg" :xl="inputItim.xl">

                    <el-form-item label="QQ 号 : " prop="qqNum">

                        <el-input v-model="ruleForm.qqNum" type="text" size="large" autocomplete="off" clearable />

                    </el-form-item>

                </el-col>

                <el-col :xs="inputItim.xs" :sm="inputItim.sm" :md="inputItim.md" :lg="inputItim.lg" :xl="inputItim.xl">

                    <el-form-item label="邮箱 : " prop="email">

                        <el-input v-model="ruleForm.email" type="text" size="large" autocomplete="off" clearable />

                    </el-form-item>

                </el-col>

                <el-col :xs="inputItim.xs" :sm="inputItim.sm" :md="inputItim.md" :lg="inputItim.lg" :xl="inputItim.xl">

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

                        <el-input placeholder="请输入验证码" v-model="ruleForm.code" type="text" size="large" autocomplete="off"

                            clearable :style="verificationCodeClass?.input" />

                        <el-button @click="sendCode()" :style="verificationCodeClass?.button" size="large">获取</el-button>

                    </el-form-item>

                </el-col>

                <el-col :xs="inputItim.xs" :sm="inputItim.sm" :md="inputItim.md" :lg="inputItim.lg" :xl="inputItim.xl">

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

                        <el-input v-model="ruleForm.password" type="password" size="large" autocomplete="off" show-password

                            clearable />

                    </el-form-item>

                </el-col>

                <el-col :xs="inputItim.xs" :sm="inputItim.sm" :md="inputItim.md" :lg="inputItim.lg" :xl="inputItim.xl">

                    <el-form-item label="再次输入密码 : " prop="password2">

                        <el-input v-model="ruleForm.password2" type="password" size="large" autocomplete="off" show-password

                            clearable />

                    </el-form-item>

                </el-col>

                <el-row justify="center" class="registerFormItem" align="middle">

                    <el-col :xs="7" :sm="5" :md="4" :lg="3" :xl="3" align="middle">

                        <el-button type="primary" size="large" tag="a" href="/" rel="noopener noreferrer"

                            style="text-decoration-line: *none*;">返回登录</el-button>

                    </el-col>

                    <el-col :xs="7" :sm="5" :md="4" :lg="3" :xl="3" align="middle">

                        <el-button @click="submitForm(registerForm)" size="large">确认注册</el-button>

                    </el-col>

                </el-row>

            </el-form>

        </el-col>

    </el-row>

</template>

<script lang="ts" setup>

import { reactive, ref, computed } from 'vue'

import type { FormInstance, FormRules } from 'element-plus'

import { ElNotification } from 'element-plus'

const registerForm = ref<FormInstance>()

const labelPosition = ref("left")

const inputItim = reactive({

    xs: 18,

    sm: 12,

    md: 9,

    lg: 7,

    xl: 7,

})

*// 获取当前屏幕宽度*

const width = ref(window.innerWidth)

*// 根据屏幕宽度，调整验证码输入框的宽度*

const verificationCodeClass = computed(() => {

    if (width.value >= 1500) {

        return {

            input: { width: "14vw" },

            button: "5vw"

        }

    } else if (width.value >= 1100) {

        return {

            input: { width: "10vw" },

            button: "5vw"

        }

    } else if (width.value > 992 && width.value < 1100) {

        return {

            input: { width: "16vw" },

            button: "5vw"

        }

    } else if (width.value > 768 && width.value <= 992) {

        return {

            input: { width: "19vw" },

            button: "5vw"

        }

    } else if (width.value > 576 && width.value <= 768) {

        return {

            input: { width: "17vw" },

            button: "5vw"

        }

    } else if (width.value > 0 && width.value <= 576) {

        return {

            input: { width: "27vw" },

            button: "5vw"

        }

    }

})

const ruleForm = reactive({

    qqNum: '',

    email: '',

    password: '',

    password2: '',

    code: ''

})

const username = (rule: any, value: any, callback: any) => {

    console.log(rule)

    console.log(value)

    console.log(callback)

    if (value === '') {

        callback(new Error('账号为空,请认真填写~'))

    }

    callback()

}

const validatePass = (rule: any, value: any, callback: any) => {

    console.log(rule)

    if (value === '') {

        callback(new Error('密码为空,请认真填写~'))

    }

    callback()

}

const validatePass2 = (rule: any, value: any, callback: any) => {

    console.log(rule)

*// 验证密码是否一致*

    if (value !== ruleForm.password) {

        callback(new Error('两次输入密码不一致!'))

    } else if (value === '') {

        callback(new Error('请再次输入密码!'))

    }

    callback()

}

const validateEmail = (rule: any, value: any, callback: any) => {

    console.log(rule)

    if (value === '') {

        callback(new Error('邮箱为空,请认真填写~'))

    }

*// 验证邮箱格式*

    const reg = /^([a-zA-Z]|[0-9])(\w|\-)+@[a-zA-Z0-9]+\.([a-zA-Z]{2,4})$/

    if (!reg.test(value)) {

        callback(new Error('邮箱格式不正确!'))

    }

    callback()

}

const validateCode = (rule: any, value: any, callback: any) => {

    console.log(rule)

    if (value === '') {

        callback(new Error('验证码为空,请认真填写~'))

    }

    callback()

}

const rules = reactive<FormRules<typeof ruleForm>>({

    qqNum: [{ validator: username, trigger: 'blur' }],

    password: [{ validator: validatePass, trigger: 'blur' }],

    password2: [{ validator: validatePass2, trigger: 'blur' }],

    email: [{ validator: validateEmail, trigger: 'blur' }],

    code: [{ validator: validateCode, trigger: 'blur' }]

})

const sendCode = () => {

*// 验证邮箱输入框规则*

    registerForm.value?.validateField('email', (valid) => {

        if (valid) {

*// TODO: 发送验证码*

            ElNotification({

                title: '提示',

                message: '验证码已发送，请注意查收',

                type: 'success',

            })

        } else {

            ElNotification({

                title: '警告',

                message: '您输入的邮箱有问题，请检查后再试',

                type: 'warning',

            })

        }

    })

}

const submitForm = (formEl: FormInstance | undefined) => {

    if (!formEl) return

    formEl.validate((valid) => {

        if (valid) {

            ElNotification({

                title: '提示',

                message: '正在注册，请稍后',

                type: 'info',

            })

*// TODO: 注册*

        } else {

            ElNotification({

                title: '警告',

                message: '请按照要求输入正确的内容',

                type: 'warning',

            })

            console.log('error submit!')

            return false

        }

    })

}

</script>

<style scoped>

*/\**

*loginForm*

*表单在屏幕中央，*

*垂直居中，*

*随着屏幕的宽度而变化*

*\*/*

.registerForm {

    justify-content: *center*;

    align-items: *center*;

    height: 75vh;

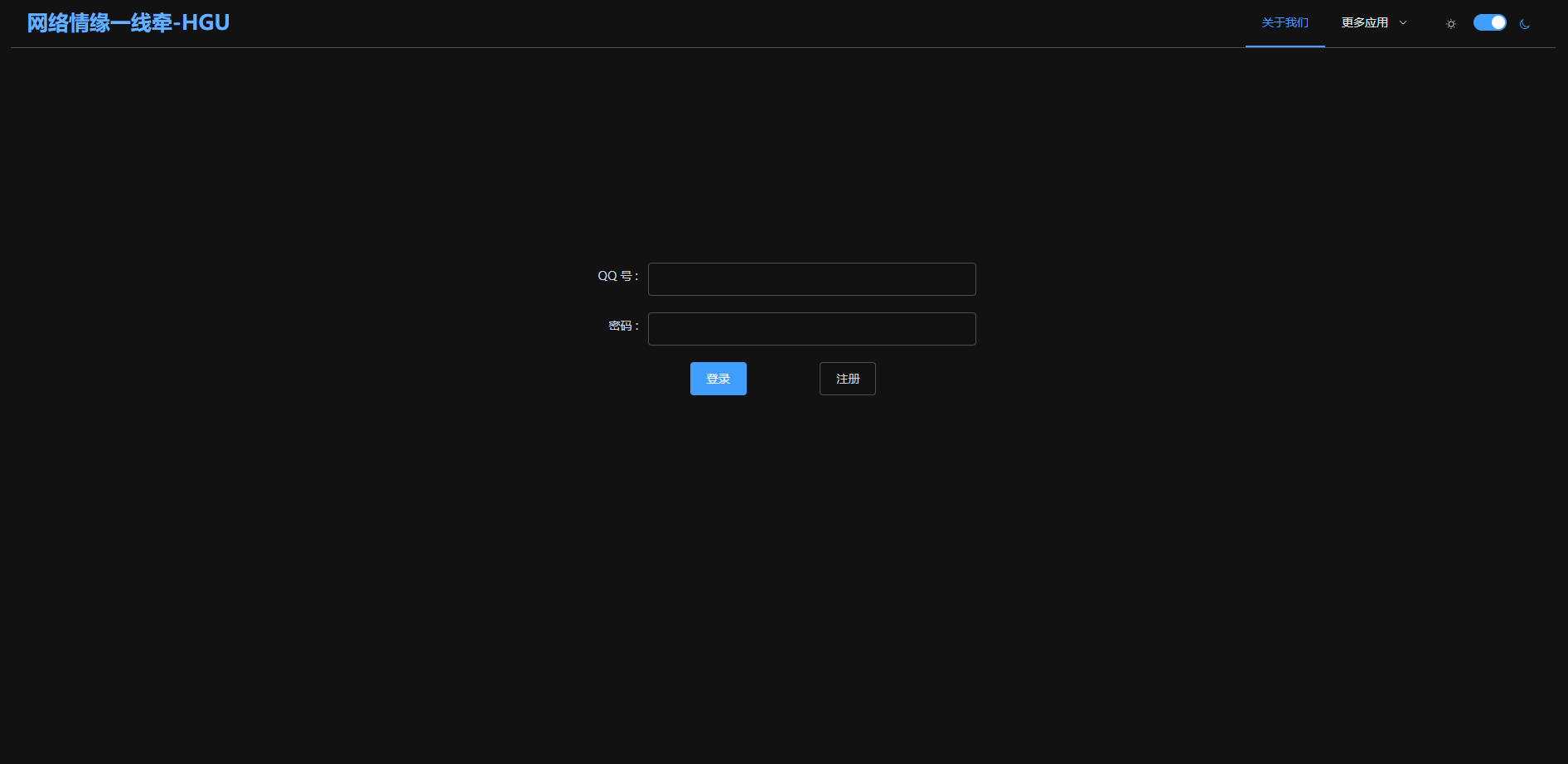
}

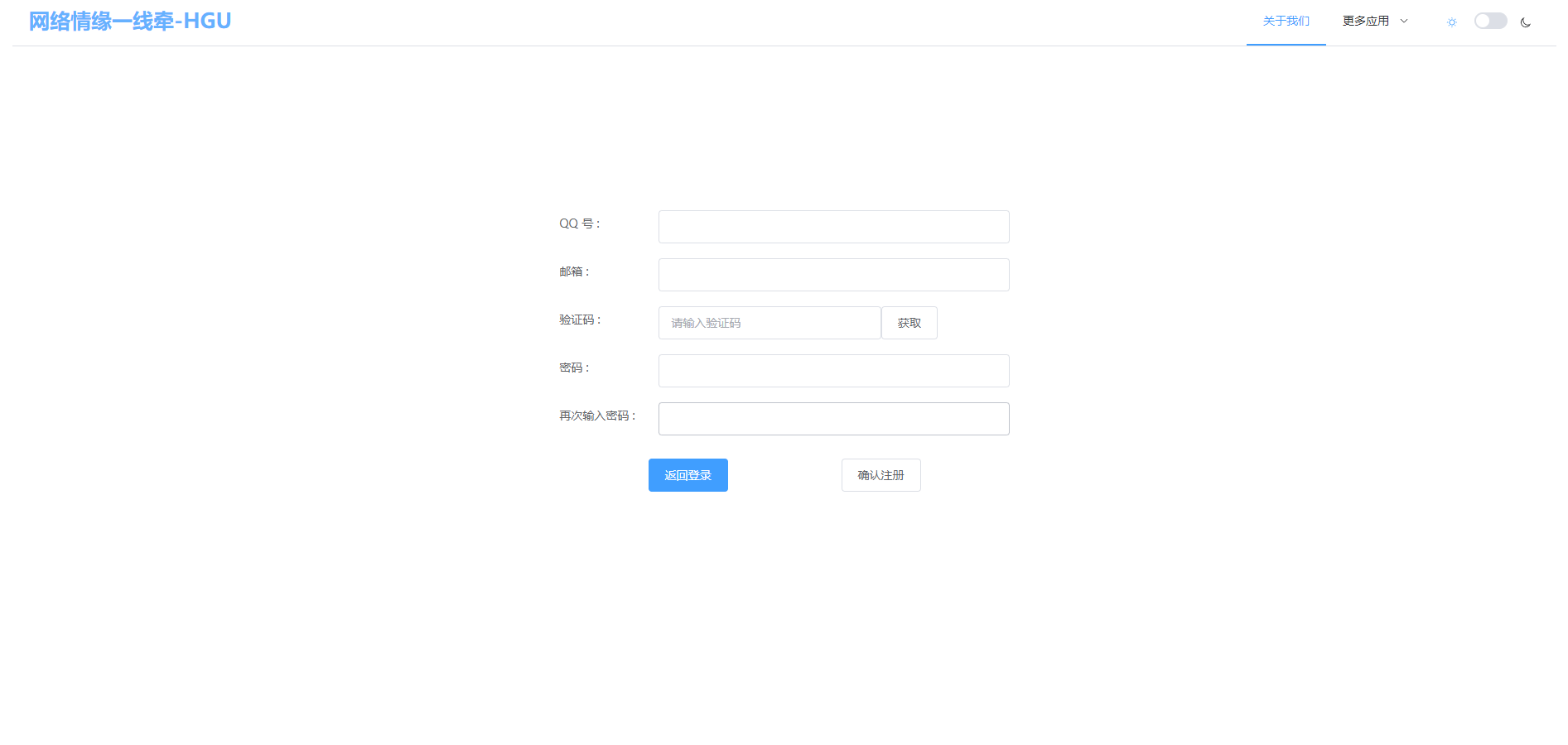
.registerFormItem {

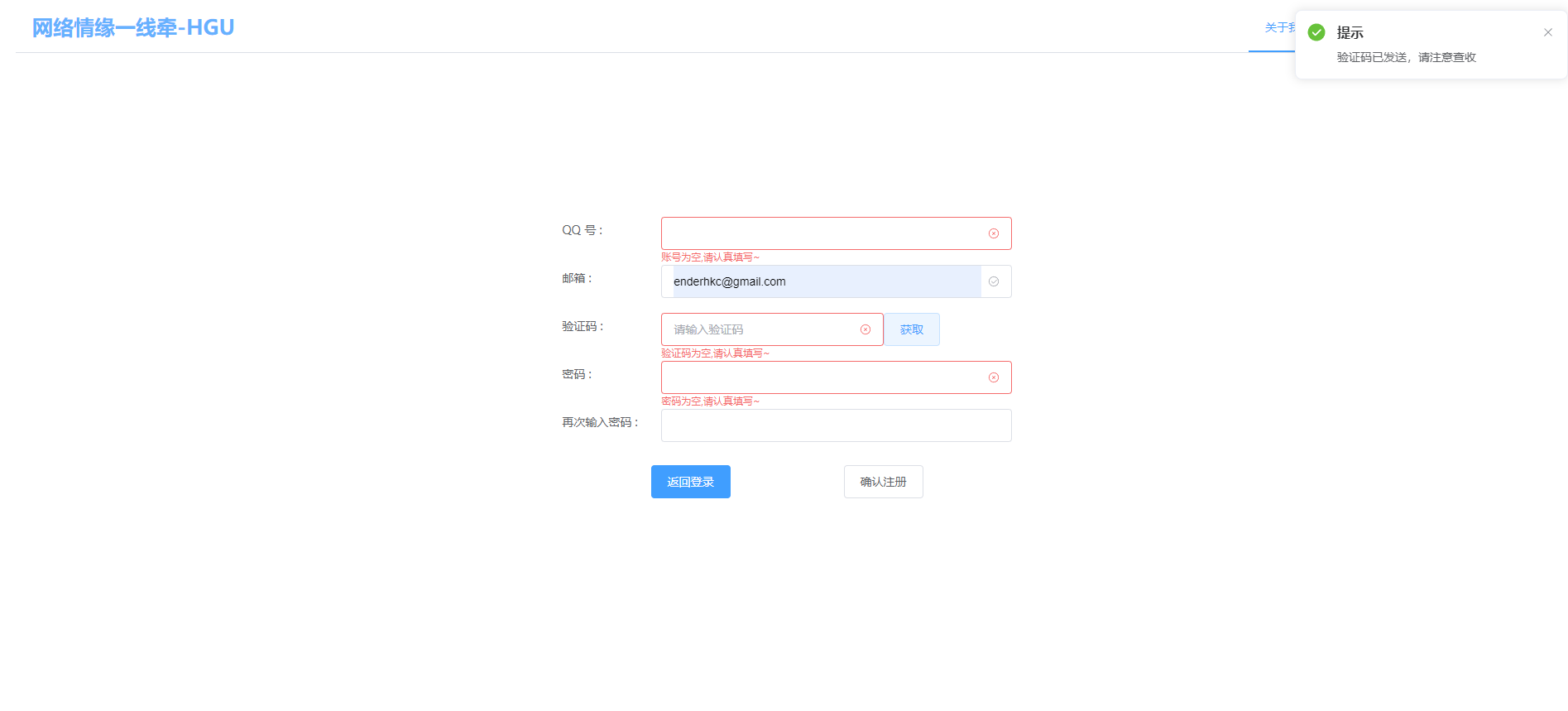
    height: 60px;

}

</style>

运行截图：





**三、心得体会：**

1. 善于利用搜索引擎

2. 熟读文档

3.熟读大佬代码，学习目录结构，代码格式规范