1. vue环境配置

安装vue脚手架

1. mysql环境

本地的mysql；

用户名：root

本地连接密码：root，注意黑马项目中的mysql密码配置的也是root

1. 出现的问题

二、实现步骤

（一）登录界面

1.布局



1. 项目架构

src中的main.js是项目的入口文件，导入vue，App组件，router路由，通过创建vue实例将App根组件渲染到了页面中，同时把路由router挂载到实例中。

App根组件中template是ui结构，script是行为，style是样式；

1. 登录组件Login.vue

vue组件都是由三部分组成， 结构，行为，样式；

注意单文件组件的样式需要用scoped属性来限制组件的样式不会扩散到全局之中，仅仅是该组件中使用该样式。

<template>

    <div>

        登录组件

    </div>

</template>

<script>

export default {

}

</script>

<style lang="less" scoped>

</style>

创建好Login.vue组件后，到router.js中导入该组件（注意导入组件的路径一定要正确），并且创建相应的路由规则，如下所示，访问的是/login,则转到Login组件中：

const routes = [

  {

    path: '/login',

    component: Login

  }

]

最后在根组件App中添加一个路由占位符，能够将所有扫描到的路由组件都显示出来。

<template>

  <div id="app">

    <h1>APP根组件</h1>

    <!-- 路由占位符 -->

    <router-view></router-view>

  </div>

</template>

当然，如果用户访问的是/根目录，则会重定向到/login页面之中，因此需要修改路由规则：

const routes = [

  {

    path: '/',

    redirect: '/login'

  },

  {

    path: '/login',

    component: Login

  }

]

less相关依赖报错，版本过高，因此使用如下的版本：

图形化界面提供的less-loader和less版本过高。---卸载上两个依赖后，在项目文件夹终端执行 npm install less-loader@4.1.0 -D | 以及 npm install less@3.9.0 -D 即可安装正常依赖

此时查看显示页面中，可以看到组件并没有撑满屏幕，因此需要设置一个全局的样式，为html,body,#app设置样式：

/\* 全局样式表 \*/

html,body,#app {

    height: 100%;

    margin: 0;

    padding: 0;

}

然后在main.js入口中导入全局样式表：

// 导入全局 样式表

import './assets/css/global.css'

创建一个登陆盒子div，并且为它设置样式，将它放到页面中间，注意看样式中使用的是绝对位置，首先基于页面左上角进行偏移50%，再进行移动；

.login\_box {

    width: 450px;

    height: 300px;

    background-color: #fff;

    border-radius: 3px;

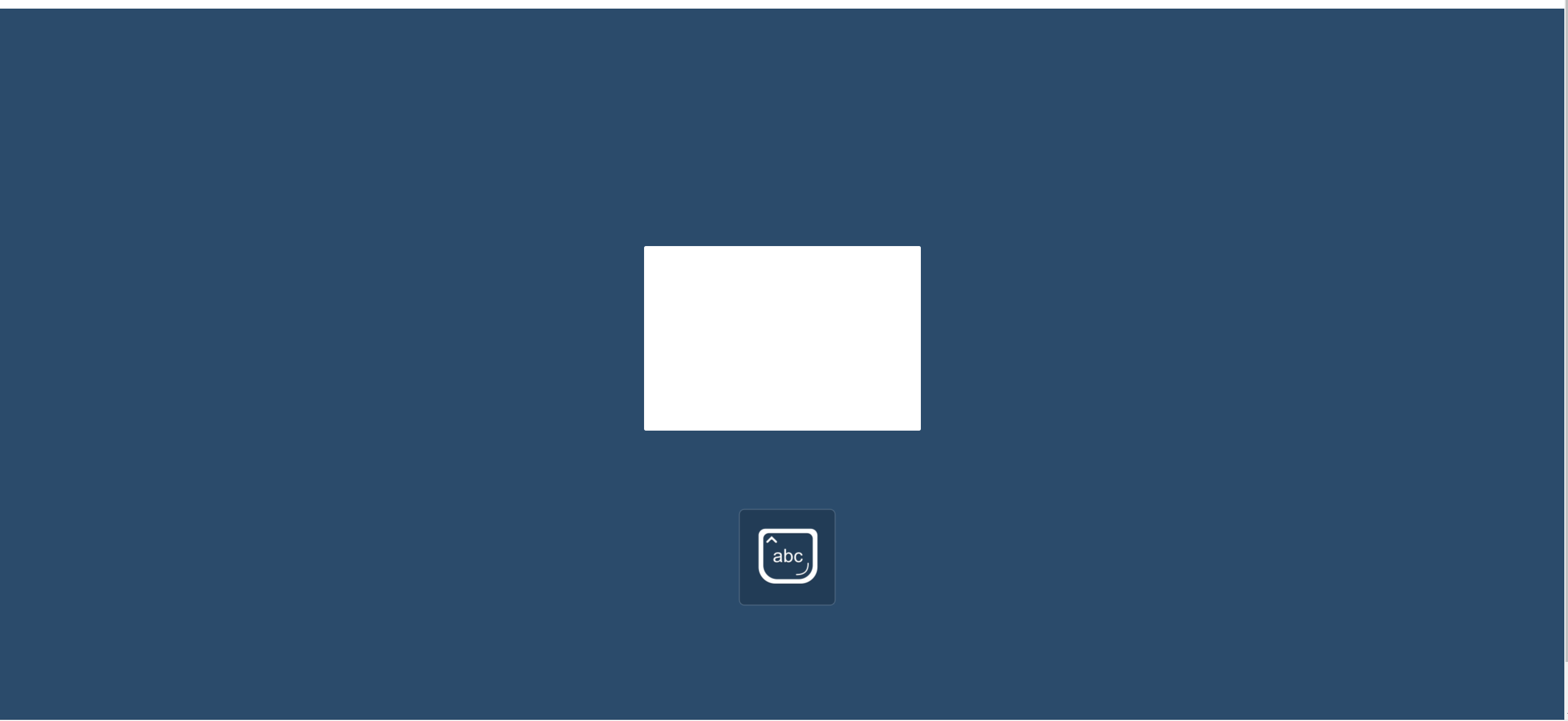
    position: absolute;

    left: 50%;

    top: 50%;

    transform: translate(-50%,-50%);

}



接下来设置图片以及样式，首先为图片在上面的盒子中再创建一个小盒子，并且撑满该子盒子，这样图片就显示在了父盒子的左上角；

然后为子盒子添加边界显示，同时为边界添加圆角样式，并且为图片设置圆角样式；

最后为图片盒子调整位置，设置绝对位置，向右以及向上偏移

.avatar\_box {

    height: 130px;

    width: 130px;

    border: 1px solid #eee;

    border-radius: 50%;

    padding: 10px;

    box-shadow: 0 0 10px #ddd;

    position: absolute;

    left: 50%;

    transform: translate(-50%, -50%);

    background-color: #fff;

    img {

        width: 100%;

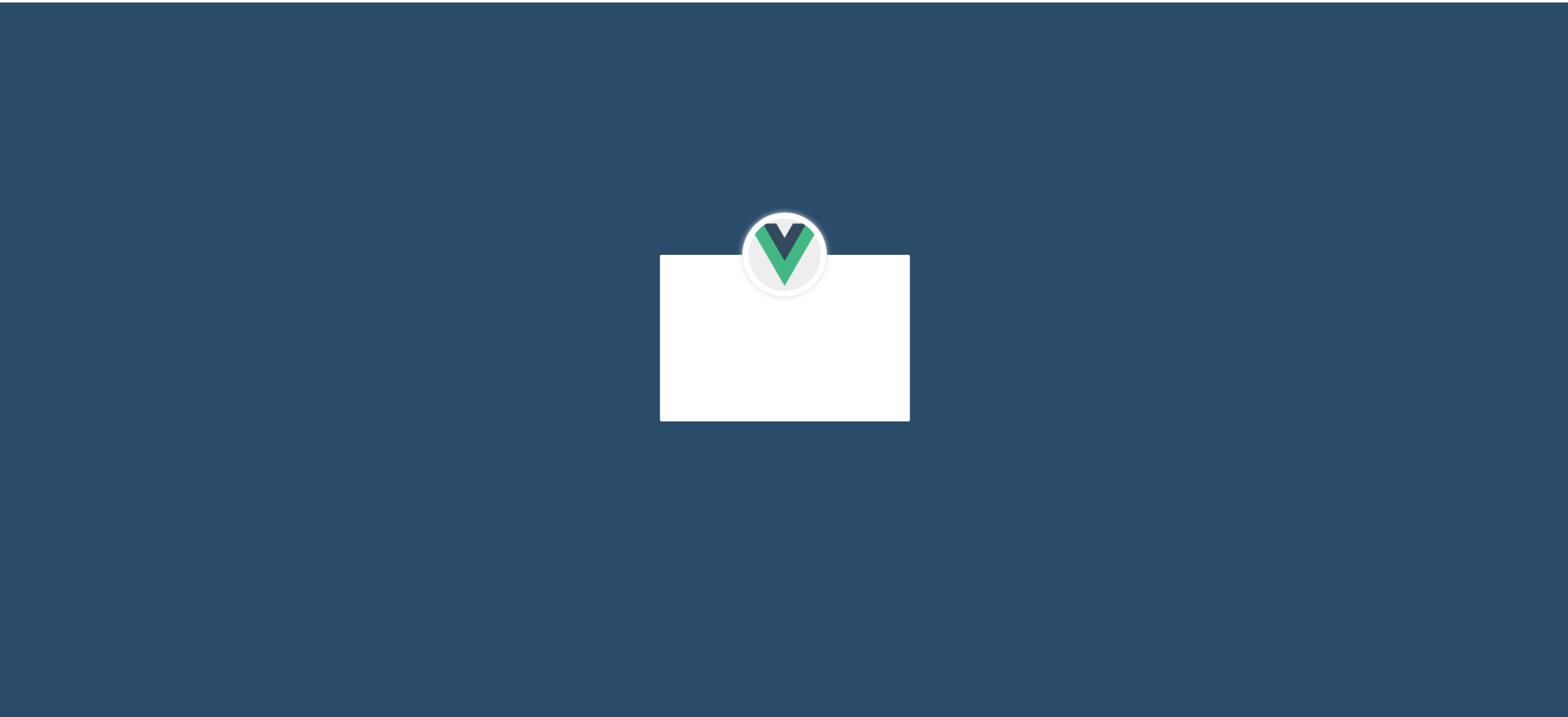
        height: 100%;

        border-radius: 50%;

        background-color: #eee;

    }

}



表单的编写，直接通过element ui进行调用，直接使用代码即可，但是会发现报错了，因为element ui是按需导入的，使用哪些el组件就导入哪些组件。因此，在element.js中进行导入：

import Vue from 'vue'

import { Button,Form,FormItem,Input } from 'element-ui'

Vue.use(Button)

Vue.use(Form)

Vue.use(FormItem)

Vue.use(Input)

接下来添加两个按钮，并且设置按钮的样式，尾部对齐：

.login\_form {

    position: absolute;

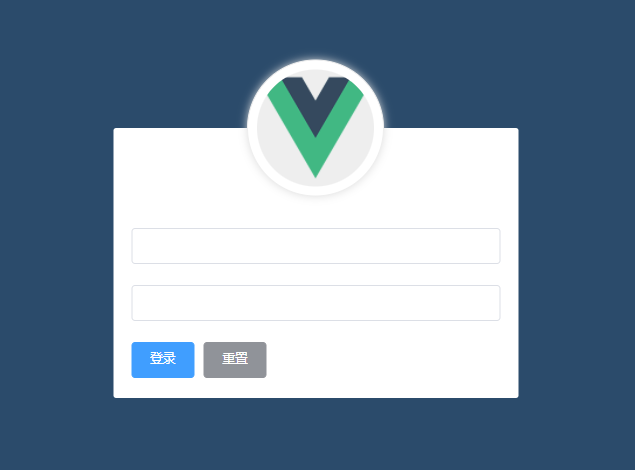
    bottom: 0;

    width: 100%;

    padding: 0 20px;

    box-sizing: border-box;

}



下面为输入框中添加小图标，通过icon属性来添加即可，从element ui组件中查询icon图标有哪些可以直接使用，或者使用第三方的图标库也行：

                <!-- 用户名 -->

                <el-form-item >

                    <el-input prefix-icon="el-icon-user"></el-input>

                </el-form-item>

                <!-- 密码 -->

                <el-form-item >

                    <el-input prefix-icon="el-icon-lock"></el-input>

                </el-form-item>



接下来为登录组件绑定数据源，也就是与后端连接。首先查看element ui官网中的form表单中的代码展示，通过:model属性（v-model）绑定一个在script中定义的数据对象，表单中的每一项都对应着该数据对象中的属性。

因此，我们需要根据表单中展示的数据定义数据对象loginForm，如username和password。然后为表单中的每一项绑定数据对象信息。

如果是密码文本框，则添加属性type=”password”即可：

            <!-- 登录表单区域 -->

            <el-form :model="loginForm" label-width="0px" class="login\_form">

                <!-- 用户名 -->

                <el-form-item >

                    <el-input v-model="loginForm.username" prefix-icon="el-icon-user"></el-input>

                </el-form-item>

                <!-- 密码 -->

                <el-form-item >

                    <el-input type="password" v-model="loginForm.password" prefix-icon="el-icon-lock"></el-input>

                </el-form-item>

                <!-- 按钮区域 -->

                <el-form-item class="btns">

                    <el-button type="primary">登录</el-button>

                    <el-button type="info">重置</el-button>

                </el-form-item>

            </el-form>

接下来对表单进行验证操作，通过element ui表单组件进行编写，直接通过一个验证规则对象就可以验证。

首先在表单区域的标签el-form标签中添加:rules绑定对象loginFormRules，然后在script中定义规则对象，分别定义验证用户名和密码的规则：

            //表单验证规则对象

            loginFormRules: {

                //验证用户名

                username: [

                    {required: true, message: "请输入登录用户名", trigger:"blur"},

                    {min: 3, max: 10, message: "长度在3-10个字符", trigger:"blur"}

                ],

                //验证密码

                password: [

                    {required: true, message: "请输入登录密码", trigger:"blur"},

                    {min: 6, max: 15, message: "长度在6-15个字符", trigger:"blur"}

                ]

            }

最后，在用户名和密码的输入框标签中绑定规则对象：

                <!-- 用户名 -->

                <el-form-item prop="username">

                    <el-input v-model="loginForm.username" prefix-icon="el-icon-user"></el-input>

                </el-form-item>

                <!-- 密码 -->

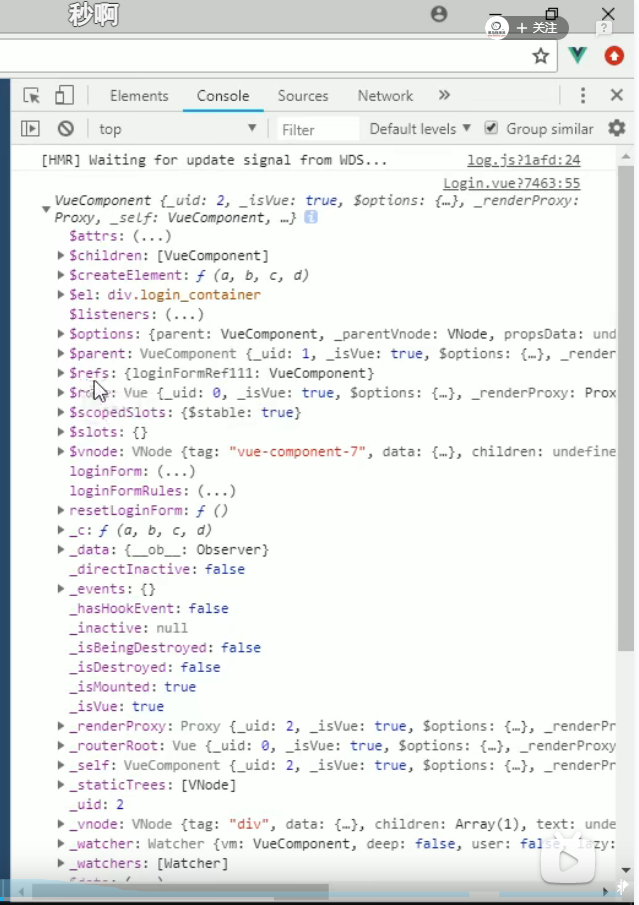
                <el-form-item prop="password">

                    <el-input type="password" v-model="loginForm.password" prefix-icon="el-icon-lock"></el-input>

                </el-form-item>

接下来，实现重置表单按钮。首先为按钮绑定重置函数，重置函数中通过获取表单对象，并且执行element ui中的form表单的函数resetFields()即可实现。

因此，需要获取表单的实例对象，通过ref标签来获取，这些都是可以通过chrome的开发者模式看到的，因此才能够通过引用对象直接获取表单实例对象。



    methods: {

        //点击重置按钮，重置登录表单

        resetLoginForm() {

            this.$refs.loginFormRef.resetFields();

        }

    }

接下来进行表单的预验证，通过调用表单的validate()函数进行校验；

首先，为登录button绑定函数login，在该函数中获取表单实例对象，调用validate()函数验证：

        //登录前的预验证

        login() {

            this.$refs.loginFormRef.validate(valid => {

                console.log(valid);

            });

        }

在验证完表单之后，需要根据验证结果发送http请求，因此在main.js中导入axios包，设置请求根路径：

//导入axios

import axios from 'axios'

//配置请求的根路径

axios.defaults.baseURL = 'http://127.0.0.1:8888/api/private/v1/'

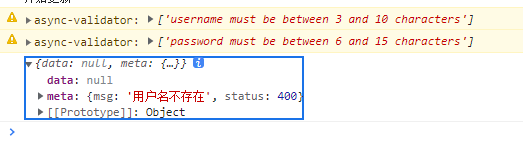
//全局属性，所有组件都可以访问

Vue.prototype.$http = axios

然后在login表单中的验证函数中进行调用post请求，参数是地址和传递数据对象；

通过f12查看返回结果，如果是promise对象，则使用await, async修饰；

从返回的data对象中直接取出res对象，也就是结果，如下所示：



        //登录前的预验证

        login() {

            this.$refs.loginFormRef.validate(async valid => {

                // console.log(valid);

                if(!valid) return;

                //获取data对象中的res

                const { data: res} = await this.$http.post('login', this.loginForm);

                console.log(res);

            });

        }

接下来进行登录结果的弹窗提示，导入element ui中的Message组件即可，在element.js中：

                if(res.meta.status != 200) return this.$message.error("登录失败");

                this.$message.success("登录成功");

登录成功后，需要记录后台返回的用户token信息，并且将token信息保存在浏览器的sessionStorage中，因为token只应该在网站打开期间有效；

同时在登录成功或失败后需要跳转到后台主页中，路由地址是/home，创建一个[Home.vue组件，代表后台主页，然后在router.js](http://Home.vue组件，代表后台主页，然后在router.js)中定义路由规则；这里由于不知道后台的密码规则，所以即使登录失败，也可以进入主页之中；

const routes = [

  {

    path: '/',

    redirect: '/login'

  },

  {

    path: '/login',

    component: Login

  },

  {

    path: '/home',

    component: Home

  }

]



接下来添加路由导航守卫控制访问权限，当用户未登陆下，直接访问有权限的页面时需要重新导航到登录页面中。

使用的是router的beforeEach方法，参数是to，from，next；要去哪个url，从哪个url来，回调函数next表示放行；

//挂载路由导航守卫

router.beforeEach((to,from,next) => {

  if(to.path == '/login') return next()

  //获取token,这里由于登录失败没有token，因此暂且不适用

  const tokenStr = window.sessionStorage.getItem('token')

  if(!tokenStr) return next('/login')

  next()

})

接下来是退出功能，基于token的退出只需要消耗本地的token即可，如果需要重新登录就生成一个新的token即可。

在点击了退出按钮后，就会清除本地的token信息，并且跳转到login页面之中；