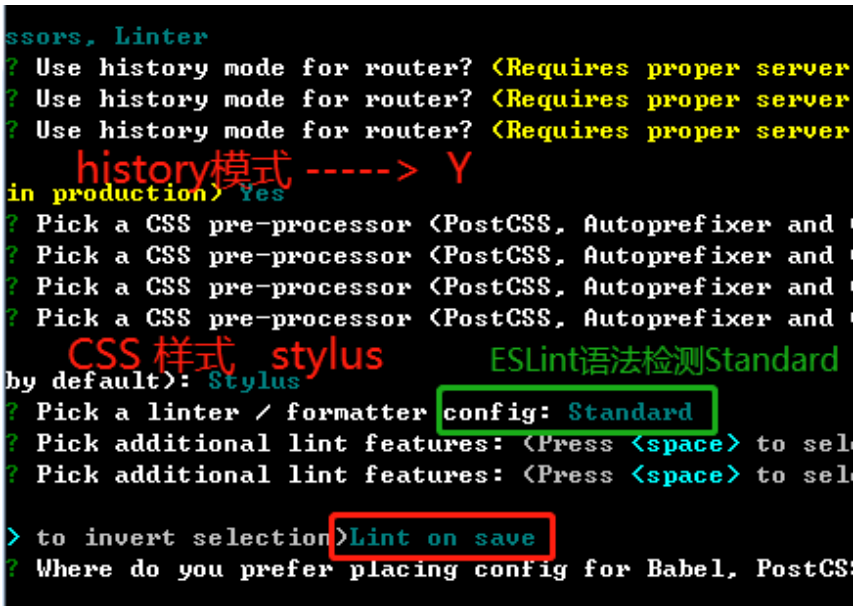
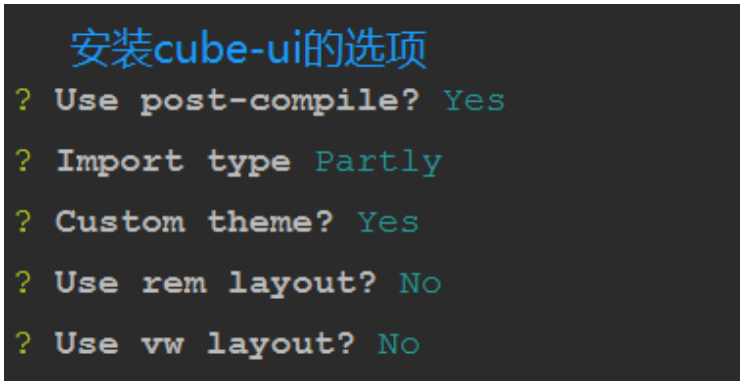
1、创建项目

选中Router和Vuex及CSS



2、使用cube-ui 移动UI库

安装 vue add cube-ui



在组件中直接使用组件库并配置相关数据，如下：

```
<div>
    <!-- cube的from表单 -->
    <cube-form
        :model="model"
        :schema="schema"
        @validate="validateHandler"
        @submit="submitHandler"></cube-form>
</div>

data(){        //   表单所需要的数据及配置
    return{
```

```
model:{
        username:"",
        passwd:""
    },
    schema:{
      fields:[{
      type:'input',
      modelKey:'username',     //  与model的username绑定
      label:'用户名',
      props:{
         placeholder:'请输入用户名',
      },
      rules:{
         required:true          // 效验为必填 不可为空
      },
         trigger:'blur'         // 效验规则为  失去焦点后
      },
      {
      type:'input',
      modelKey:'passwd',        //  与model的passwd绑定
      label:'密码',
      props:{
         type:'password',       //  输入框的类型为密码
         placeholder:'请输入密码',
         eye:{
           open:false           //  输入密码默认不可见为*号
      }
    },
      rules:{
         required:true
      },
      trigger:'blur'
    },
    {
      type:'submit',        //  提交按钮
```

```
            label:'登录'
          }
        ]
    }
  }
},
cube-ui组件的相关事件处理
methods:{
    validateHandler(ret){

    },
    async submitHandler(e){
       e.preventDefault()  // 阻止表单自动提交
     /*  为什么 这么传数据
         后端要的数据格式是
         name: xxxm
         pwd: 123
        所以要对数据做处理
        而data:{
           name:this.model.username,
           pwd:this.model.passwd
        }   这种方式给后端传数据的格式是:
        {name: xxxm, pwd: 123}的一个对象格式
     */
   //对post请求的数据处理
   let param = new URLSearchParams();
   param.append("name", this.model.username);
   param.append("pwd", this.model.passwd);
   //发起网络请求获取结果
   const rets = await this.$axios({
       method:"post",
       url:"http://localhost:8080/Antes/api/customer/UserKey",
       data:param
       })
        if (rets.code==1&&rets.success){
```

```
            //  存token  将token持久化存入localStorage里（浏览器中存储）
            localStorage.setItem('token',rets.token)
            //   修改vuex中的数据值
            this.$store.commit('settoken',rets.token)
            //登录成功清空输入框
            this.model.username=""
            this.model.passwd=""
            // cube-ui的提示 如android的toast一样
            const toast = this.$createToast({
                time: 3000,
                txt: rets.message,
                type:"correct"
            })
            toast.show()
        }
    }
}
```

网络请求： 需要安装**axios**并在**main**中引入设置为总线模式

```
import axios from 'axios'

Vue.prototype.$axios = axios
```

axios的**post**请求时数据的处理要注意

```
 // 这种方式获取的数据传给后端的是一个对象：{name: xxxm, pwd: 123}
 data:{
   name:this.model.username,
   pwd:this.model.passwd
}
// 这种方式封装的数据是单独的数据对象：  name: xxxm    pwd: 123
let param = new URLSearchParams();
param.append("name", this.model.username);
param.append("pwd", this.model.passwd);
```

```
data:param
```

localStorage持久化数据到浏览器中

通过localStorage可将某些数据存入到浏览器中，如：用户登录成功的 token 值

```
localStorage.setItem('token',rets.token)
清空localStorage 浏览器存储
localStorage.removeItem('token')
从浏览器中获取存入的数据
const token = localStorage.getItem('token')
```

http的请求拦截与服务器响应拦截

每次请求网络都在请求头中带着自定义的请求头数据去服务器请求

1、新建一个setaxios的js文件

2、引入相关库

```
import axios from 'axios'
import store from './store'
import router from './router'
```

3、定义请求拦截和响应拦截

```
每次请求网络都带上token
export default function setAxios() {
    axios.interceptors.request.use(
        config=>{
            if (store.state.token){
                config.headers.token = store.state.token
            }
            return config
        }
    )
//获取数据的时候，如果code响应为-1  就当做是登录过期了，需要重新登录
//每一次http请求有返回，都先触发这个拦截器
    axios.interceptors.response.use(
```

```
        response=>{
            if (response.status==200){
                const data = response.data
                if (data.code==-1){
                    //清空vuex中token的值
                    store.commit('settoken','')
                    //清空localStorage 浏览器存储
                    localStorage.removeItem('token')
                    //跳转
                    router.replace({
                        path:'login'
                    })
                }
                return data
            }
            return response
        }
    )
}
```

4、在main.js中注册引用

```
import setAxios from './setaxios'
setAxios()
```

路由配置中的标记字段meta

```
{
  path: '/about',
  name: 'about',
    //添加标记，拦截跳转需要登录
  meta:{
    auth:true
  },
```

以上路由中定义了标记 那么在路由全局守卫中可以根据标记来限制访问

```
路由的钩子   路由守卫
```

```
router.beforeEach((to,from, next)=>{
  if(to.meta.auth){
    if(store.state.token){
      next()
    } else{
      next({
          path:'/login'
      })
    }
  }else{
    next()
  }
})
```