电商项目实战01

选择一个合适的UI库

	Mint-ui	vux	Cube-ui	vant
维护团队	饿了么团队	个人	滴滴	有赞
更新迭代	1年没提交	三个月	持续更新	持续更新
Github	13K	14k	5K	7K
测试	基本没啥覆盖	基本没覆盖	测试完备	测试完备 80%
缺点	基本停止维护	个人项目 更新迭代不及时 没适配vue-cli3		
特点			后编译，create-api，滴滴线上业务	有赞线上项目，支持SSR，TS

```
vue add cube-ui
```

```
added 5 packages from 4 contributors in 6.276s
✓  Successfully invoked generator for plugin: vue-cli-plugin-cube-ui
   The following files have been updated / added:

     src/cube-ui.js
     src/theme.styl
     vue.config.js
     package-lock.json
     package.json
     src/main.js
```

扩展性

任何UI库 都不能满足全部的业务开发需求，都需要自己进行定制和扩展， 组件化设计思路至关重要

登录页面

```
let router = new Router({
  mode: 'history',
  base: process.env.BASE_URL,
  routes: [
    {
      path: '/',
      name: 'home',
      component: Home
    },
    {
      path:"/login",
      name:'login',
      component:Login
    },
    {
      path: '/about',
      name: 'about',
      meta:{
        // 标记需要登录
        auth:true
      },
      // route level code-splitting
      // this generates a separate chunk (about.[hash].js) for this route
      // which is lazy-loaded when the route is visited.
      component: () => import(/* webpackChunkName: "about" */ './views/About.vue')
    }
  ]
})
```

登录表单

cube-ui的表单文档

```
<div>
  <div class="logo">
    <img src="https://img.kaikeba.com/logo-new.png" alt="">
  </div>
  <!-- <cube-button>登录</cube-button> -->
  <cube-form
    :model="model"
    :schema="schema"
    @submit="handleLogin"
    @validate="haneldValidate"
  >
  </cube-form>
</div>
```

分别设置model和schema

```
 data() {
    return {
      model: {
        username: "",
        passwd: ""
      },
      schema: {
        fields: [
          {
            type: "input",
            modelKey: "username",
            label: "用户名",
            props: {
              placeholder: "请输入用户名"
            },
            rules: {
              // 校验规则
              required: true
            },
            trigger: "blur"
          },
          {
            type: "input",
            modelKey: "passwd",
            label: "密码",
            props: {
              type: "password",
              placeholder: "请输入密码",
              eye: {
                open: true
              }
            },
```

```
            rules: {
              required: true
            },
            trigger: "blur"
          },
          {
            type: "submit",
            label: "登录"
          }
        ]
      }
    };
  }
```

model就是输入框绑定的数据，schema是具体的表单的描述，会动态渲染一个表单，里面的rule配置和validator组件 保持一致

每次校验都会除法handleValidate方法，打印校验的结果

```
haneldValidate(ret){
  console.log(ret)
},
```

点击登录除法handleLogin 发起登录请求

```
async handleLogin(e){
  // 阻止表单自动提交
  e.preventDefault()
  const obj = {
    username:this.model.username,
    passwd:this.model.passwd
  }
  const ret = await axios.get('/api/login',{params:obj})
  // if(ret.s)
  console.log(ret)
}
```

vue.config.js中，模拟接口，设置kaikeba和123为合法用户名和密码

```
app.get("/api/login", function(req,res){
  const {username, passwd} = req.query
  if(username=='kaikeba' && passwd =='123'){
    res.json({
      code:0,
      token:'kaiekbazhenbucuo-'+(new Date().getTime()+1000*60)
    })
  }else{
    res.json({
      code:1,
      message:"用户名或者密码错误"
    })
  }
```

```
})
```

返回的token是一个随机字符串和过期事件，实际开发中随机字符串要确保足够复杂

前端登录失败 使用toast弹出报错，这里用到了cube-ui一个非常优秀的api： create-api，任何组件都可以使用javascript调用

登录成功后，存储token 并且提交至vuex里

```
if(ret.code==0){
    localStorage.setItem('token',ret.token)
    this.$store.commit('settoken',ret.token)
    // 写token的逻辑
}else{
    const toast = this.$createToast({
        time:2000,
        txt:ret.message||'未知错误',
        type:'error'
    })
    toast.show()
}
```

设置vuex

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({
  state: {
    token:""
  },
  mutations: {
    settoken(state,token){
      state.token = token
    }
  },
  actions: {

  }
})
```

http拦截器

有了token之后，每次http请求发出，都要加载header上

```
axios.interceptors.request.use(
config=>{
  if(store.state.token){
    config.headers.token = store.state.token
  }
  return config
}
)
```

并且初始化App.vue的时候，需要把本地存储里的token 提交至vuex

```
async created(){
const token = localStorage.getItem('token')
if(token){
  this.$store.commit('settoken',token)
}
}
```

注销

无论是主动触发logout 还是token自动过期，都约定返回码是-1的时候是注销token的操作

```
app.get("/api/logout", function(req,res){
   res.json({
       code:-1
   })
})
```

http拦截响应

统一处理code是-1的情况，清理token 跳转login

```
axios.interceptors.response.use(
response=>{
  if(response.status==200){
    const data = response.data
    if(data.code==-1){
      // 登录过期了
      // 注销登录 清空vuex和localstorage
      store.commit('settoken','')
      localStorage.removeItem('token')
      // 跳转login
      router.replace({
        path:'login'
      })
      // store.commit(LOGOUT,'')
```

```
    }
    return data
  }
  return response
}
)
```

作业

1. /api/goods 等数据接口 token过期做拦截 如果token时间过期 自动清理token
2. https://didi.github.io/cube-ui/#/zh-CN/docs/validator
    1. 根据cube-ui的校验写法，对用户名做异步校验 如果是dasheng，提示用户名不存在
    2. 校验用户名通过网络接口 返回用户是否可以登录