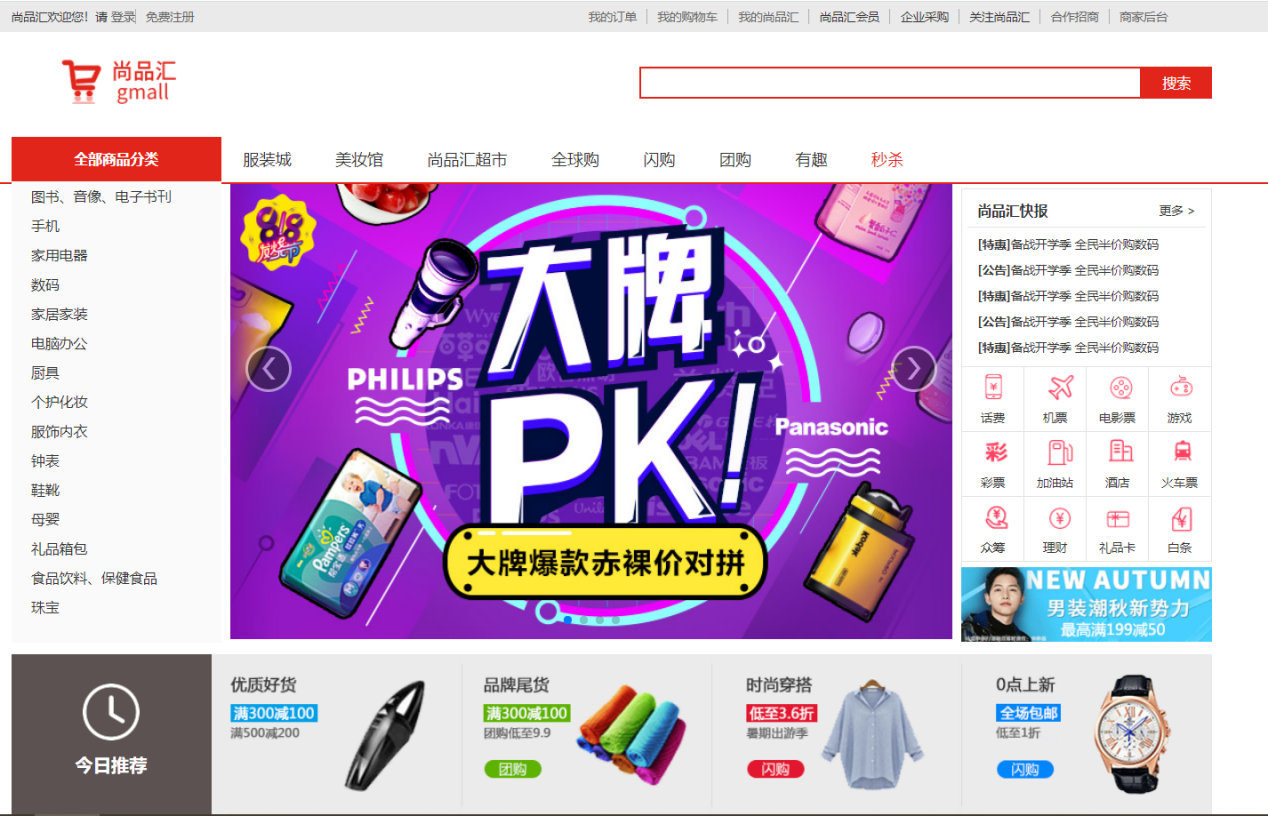
Vue电商项目-前台PC

# 第1章: 准备

## 项目描述(面试问项目的第一个问题)

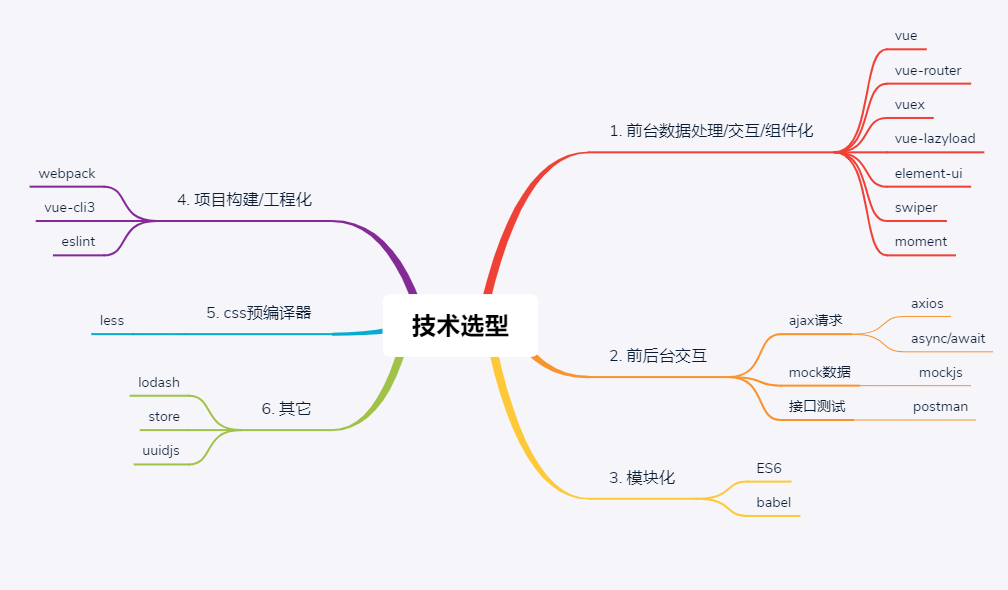
1. 此项目为在线电商Web App (SPA)
2. 包括首页, 搜索列表, 商品详情, 购物车, 订单, 支付, 用户登陆/注册等多个子模块
3. 使用Vue全家桶+ES6++Webpack+Axios等前端最新最热的技术
4. 采用模块化、组件化、工程化的模式开发

## 项目功能界面



说明: 完整功能界面运行最终版项目

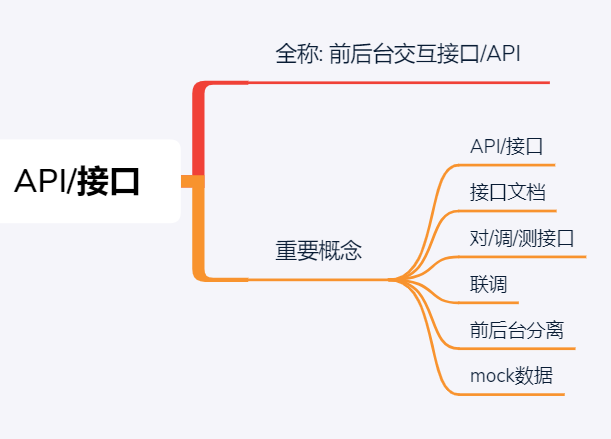
## 技术选型



## 前端路由



## API/接口



## 你能从此项目中学到什么?

### 流程及开发方法

1. 熟悉一个项目的开发流程
2. 学会模块化、组件化、工程化的开发模式
3. 掌握使用vue-cli脚手架初始化Vue.js项目
4. 学会模拟json后端数据，实现前后端分离开发
5. 学会ES6+eslint的开发方式
6. 掌握一些项目优化技巧

### Vue插件或第三方库

1. 学会使用vue-router开发单页应用
2. 学会使用axios与后端进行数据交互
3. 学会使用vuex管理应用组件状态
4. 学会使用swiper实现页面滑动效果
5. 学会使用elment-ui组件库构建界面
6. 学会使用vue-lazyload实现图片惰加载
7. 学会使用mockjs模拟后台数据接口

# 第2章: 应用开发详解

## 开启项目开发

### 使用Vue CLI 3(脚手架)搭建项目

1. Vue CLI是vue官方提供的用于搭建基于vue+webpack+es6项目的脚手架工具
2. 在线文档: [<https://cli.vuejs.org/zh/>](https://github.com/vuejs/vue-cli)
3. 操作:

**# 使用vue-cli3**

npm install -g @vue/cli

vue create shop-client

cd shop-client

npm run serve

**# 降级到vue-cli2**

npm install -g @vue/cli-init

vue init webpack gshop-client2

cd shop-client

npm run dev

### 项目结构分析

shop-client

|-- node\_modules

|-- public

|-- index.html: 主页面文件

|-- src

|-- main.js: 应用入口js

|-- babel.config.js: babel的配置文件

|-- vue.config.js: vue的配置文件

|-- .gitignore: git版本管制忽略的配置

|-- package.json: 应用包配置文件

|-- README.md: 应用描述说明的readme文件

### 编码测试与打包发布项目

1. 编码测试

npm run serve

访问: <http://localhost:8080>

编码, 自动编译打包(HMR), 查看效果

1. 打包发布

npm run build

npm install -g serve

serve dist -p 5000

访问: [http://localhost:5000](http://localhost:9000)

## 项目源码目录设计



## ESLint

### 理解

1. ESLint是一个代码规范检查工具
2. 基本已替代以前的JSHint

### ESLint提供以下支持

1. ES6
2. JSX
3. 自定义错误和提示

### ESLint提供以下几种校验

1. 语法错误校验
2. 不重要或丢失的标点符号，如分号
3. 没法运行到的代码块
4. 未被使用的参数提醒
5. 漏掉的结束符，如}
6. 检查变量的命名

### 规则的错误等级有三种

1. 0：关闭规则检查 off。
2. 1：打开规则检查，并且作为一个警告（输出提示文本黄色）warn。
3. 2：打开规则检查，并且作为一个错误（输出提示文本红色）error。

### 相关配置

1. package.json : 全局规则配置文件

'rules': {

'no-new': 'off'

}

1. 在js/vue文件中修改局部规则

/\* eslint-disable no-new \*/

new Vue({

el: 'body',

components: { App }

})

1. vue.config.js: 关闭规则检查

// 关闭ESLint的规则

lintOnSave: false,

## 引入vue-router

### 下载依赖包

**npm install -S vue-router**

### 编码

1. pages/Home/index.vue

|  |
| --- |
| <template>    <div>Home</div>  </template>  <script>  export default {    name: 'Home',    data () {      return {}    },  }  </script>  <style lang="less" scoped>  </style> |

1. pages/Search/index.vue

|  |
| --- |
| <template>    <div>Search</div>  </template>  <script>  export default {    name: 'Search',    data () {      return {}    },  }  </script>  <style lang="less" scoped>  </style> |

1. pages/Register/index.vue

|  |
| --- |
| <template>    <div>Register</div>  </template>  <script>  export default {    name: 'Register',    data () {      return {}    },  }  </script>  <style  lang="less" scoped>  </style> |

1. pages/Login/index.vue

|  |
| --- |
| <template>    <div>Login</div>  </template>  <script>  export default {    name: 'Login',    data () {      return {}    },  }  </script>  <style  lang="less" scoped>  </style> |

1. router/routes.js

|  |
| --- |
| import Home from '@/pages/Home'  import Search from '@/pages/Search'  import Register from '@/pages/Register'  import Login from '@/pages/Login'  /\*  所有静态路由配置的数组  \*/  export default [    {      path: '/',      component: Home    },    {      path: '/search',      component: Search    },    {      path: '/register',      component: Register    },    {      path: '/login',      component: Login    }  ] |

1. router/index.js

|  |
| --- |
| import Vue from 'vue'  import VueRouter from 'vue-router'  import routes from './routes'  // 声明使用插件  Vue.use(VueRouter)  // 向外默认暴露路由器对象  export default new VueRouter({    mode: 'history', // 没有#的模式    routes, // 注册所有路由  }) |

1. main.js

|  |
| --- |
| import Vue from 'vue'  import App from './App.vue'  import router from './router'  Vue.config.productionTip = false  new Vue({    render: h => h(App),    router, // 注册路由器  }).$mount('#app') |

1. components/Header/index.vue

|  |
| --- |
| <template>    <div>Header</div>  </template>  <script>  export default {    name: 'Header',    data () {      return {}    },  }  </script>  <style  lang="less" scoped>  </style> |

1. components/Footer/index.vue

|  |
| --- |
| <template>    <div>Footer</div>  </template>  <script>  export default {    name: 'Footer',    data () {      return {}    },  }  </script>  <style  lang="less" scoped>  </style> |

1. App.vue

|  |
| --- |
| <template>    <div>      <Header/>      <router-view></router-view>      <Footer/>    </div>  </template>  <script>  import Header from './components/Header'  import Footer from './components/Footer'  export default {    name: 'App',    components: {      Header,      Footer    }  }  </script>  <style lang="less" scoped>  </style> |

1. public/css/reset.css

|  |
| --- |
| /\* 清除内外边距 \*/  body, h1, h2, h3, h4, h5, h6, hr, p, blockquote,  dl, dt, dd, ul, ol, li,  pre,  fieldset, lengend, button, input, textarea,  th, td {      margin: 0;      padding: 0;  }  /\* 设置默认字体 \*/  body,  button, input, select, textarea { /\* for ie \*/      /\*font: 12px/1 Tahoma, Helvetica, Arial, "宋体", sans-serif;\*/      font: 12px/1.3 "Microsoft YaHei",Tahoma, Helvetica, Arial, "\5b8b\4f53", sans-serif; /\* 用 ascii 字符表示，使得在任何编码下都无问题 \*/      color: #333;  }  h1 { font-size: 18px; /\* 18px / 12px = 1.5 \*/ }  h2 { font-size: 16px; }  h3 { font-size: 14px; }  h4, h5, h6 { font-size: 100%; }  address, cite, dfn, em, var, i{ font-style: normal; } /\* 将斜体扶正 \*/  b, strong{ font-weight: normal; } /\* 将粗体扶细 \*/  code, kbd, pre, samp, tt { font-family: "Courier New", Courier, monospace; } /\* 统一等宽字体 \*/  small { font-size: 12px; } /\* 小于 12px 的中文很难阅读，让 small 正常化 \*/  /\* 重置列表元素 \*/  ul, ol { list-style: none; }  /\* 重置文本格式元素 \*/  a { text-decoration: none; color: #666;}  /\* 重置表单元素 \*/  legend { color: #000; } /\* for ie6 \*/  fieldset, img { border: none; }  button, input, select, textarea {      font-size: 100%; /\* 使得表单元素在 ie 下能继承字体大小 \*/  }  /\* 重置表格元素 \*/  table {      border-collapse: collapse;      border-spacing: 0;  }  /\* 重置 hr \*/  hr {      border: none;      height: 1px;  }  .clearFix::after{      content:"";      display: block;      clear:both;  }  /\* 让非ie浏览器默认也显示垂直滚动条，防止因滚动条引起的闪烁 \*/  html { overflow-y: scroll; }  a:link:hover{      color : rgb(79, 76, 212) !important;      text-decoration: underline;  }  /\* 清除浮动 \*/  .clearfix::after {      display: block;      height: 0;      content: "";      clear: both;      visibility: hidden;  } |

1. public/index.html

|  |
| --- |
| <link rel="stylesheet" href="/css/reset.css">  <div id="app"></div> |

### 运行, 并请求不同路由路径

查看运行效果

## Vue组件化

### 配置vue组件文件模板

|  |
| --- |
| {    "Print to vue": {      "prefix": "vue",      "body": [       "<template>",       " <div></div>",       "</template>",       "",       "<script>",       "export default {",       " name: '',",       "}",       "</script>",       "",       "<style lang=\"less\" scoped>",       "",       "</style>",       ""      ],      "description": "快速创建vue单文件组件"     }  } |

### App.vue

|  |
| --- |
| <template>    <div>      App    </div>  </template>  <script>  export default {    name: 'App'  }  </script>  <style>  </style> |

### main.js

|  |
| --- |
| **import** Vue **from 'vue' import App** **from './App'**  **new** Vue({  **el**: **'#app'**,  render: h => h(App) }) |

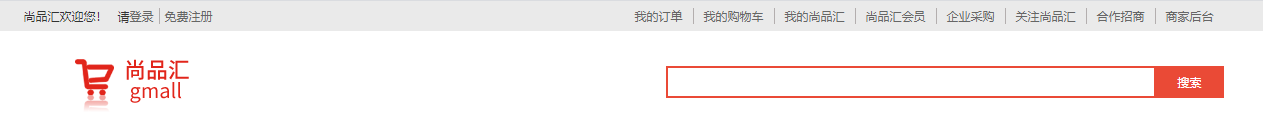
### jsconfig.json

让vscode提示@开头的模块路径引入

|  |
| --- |
| {    "compilerOptions": {      "baseUrl": "./",      "paths": {          "@/\*": ["src/\*"]      }    },    "exclude": ["node\_modules", "dist"]  } |

## Header组件

### 说明



1. 使用声明式路由导航与编程式路由导航
2. 解决编程式路由导航的一个错误

### Header组件编码

1. 准备图片资源

logo.png

1. 静态页面

|  |
| --- |
| <header class="header">    <!-- 头部的第一行 -->    <div class="top">      <div class="container">        <div class="loginList">          <p>尚品汇欢迎您！</p>          <p>            <span>请</span>            <a href="###">登录</a>            <a href="###" class="register">免费注册</a>          </p>        </div>        <div class="typeList">          <a href="###">我的订单</a>          <a href="###">我的购物车</a>          <a href="###">我的尚品汇</a>          <a href="###">尚品汇会员</a>          <a href="###">企业采购</a>          <a href="###">关注尚品汇</a>          <a href="###">合作招商</a>          <a href="###">商家后台</a>        </div>      </div>    </div>    <!--头部第二行 搜索区域-->    <div class="bottom">      <h1 class="logoArea">        <a class="logo" title="尚品汇" href="###" target="\_blank">          <img src="./images/logo.png" alt />        </a>      </h1>      <div class="searchArea">        <form action="###" class="searchForm">          <input type="text" id="autocomplete" class="input-error input-xxlarge" />          <button class="sui-btn btn-xlarge btn-danger" type="button">搜索</button>        </form>      </div>    </div>  </header> |

1. style

|  |
| --- |
| .header {    &>.top {      background-color: #eaeaea;      height: 30px;      line-height: 30px;      .container {        width: 1200px;        margin: 0 auto;        overflow: hidden;        .loginList {          float: left;          p {            float: left;            margin-right: 10px;            .register {              border-left: 1px solid #b3aeae;              padding: 0 5px;              margin-left: 5px;            }          }        }        .typeList {          float: right;          a {            padding: 0 10px;            &+a {              border-left: 1px solid #b3aeae;            }          }        }      }    }    &>.bottom {      width: 1200px;      margin: 0 auto;      overflow: hidden;      .logoArea {        float: left;        .logo {          img {            width: 175px;            margin: 25px 45px;          }        }      }      .searchArea {        float: right;        margin-top: 35px;        .searchForm {          overflow: hidden;          input {            box-sizing: border-box;            width: 490px;            height: 32px;            padding: 0px 4px;            border: 2px solid #ea4a36;            float: left;            &:focus {              outline: none;            }          }          button {            height: 32px;            width: 68px;            background-color: #ea4a36;            border: none;            color: #fff;            float: left;            cursor: pointer;            &:focus {              outline: none;            }          }        }      }    }  } |

1. template

|  |
| --- |
| <template>    <header class="header">      <!-- 头部的第一行 -->      <div class="top">        <div class="container">          <div class="loginList">            <p>尚品汇欢迎您！</p>            <p>              <span>请</span>              <router-link to="/login">登陆</router-link>              <router-link class="register" to="/register">免费注册</router-link>            </p>          </div>          <div class="typeList">            <a href="javascript:">我的订单</a>            <a href="javascript:">我的购物车</a>            <a href="javascript:">我的尚品汇</a>            <a href="javascript:">尚品汇会员</a>            <a href="javascript:">企业采购</a>            <a href="javascript:">关注尚品汇</a>            <a href="javascript:">合作招商</a>            <a href="javascript:">商家后台</a>          </div>        </div>      </div>      <!--头部第二行 搜索区域-->      <div class="bottom">        <h1 class="logoArea">          <router-link class="logo" to="/">            <img src="./images/logo.png" alt="logo" />          </router-link>        </h1>        <div class="searchArea">          <form class="searchForm">            <input type="text" id="autocomplete" class="input-error input-xxlarge" v-model="keyword"/>            <button class="sui-btn btn-xlarge btn-danger" type="button" @click="search">搜索</button>          </form>        </div>      </div>    </header>  </template> |

1. script

|  |
| --- |
| <script>    export default {      name: "Header",      data() {        return {          keyword: ''        }      },      methods: {        search () {          this.$router.push(`/search/${this.keyword}`)        }      }    }  </script> |

### router/routes.js

**问题: 如何实现params参数可传可不传?**

|  |
| --- |
| {      path: '/search/:keyword?', // params参数可传可不传      component: Search  }, |

### Search组件编码

|  |
| --- |
| <div>搜索关键字: {{$route.params.keyword}}</div> |

### 路由跳转与传参相关问题

1. 跳转路由的2种基本方式

声明式: <router-link to="">

编程式: this.$router.push()/replace()

1. 跳转路由携带参数的2种方式

params参数

query参数

1. 面试问题1:

描述: 编程式路由跳转到当前路由(参数不变), 会抛出NavigationDuplicated的警告错误

解决1: 在跳转时指定成功或失败的回调函数, 通过catch处理错误

解决2: 修正Vue原型上的push和replace方法 (优秀)

1. 面试问题2: 如何指定params参数可传可不传?

path: '/search/:keyword?'

1. 面试问题3: 指定params参数时可不可以用path和params配置的组合?

不可以用path和params配置的组合, 只能用name和params配置的组合

query配置可以与path或name进行组合使用

1. 面试问题4: 如果指定name与params配置, 但params中数据是一个"", 无法跳转

解决1: 不指定params

解决2: 指定params参数值为undefined

1. 面试问题5: 路由组件能不能传递props数据?

可以: 可以将query或且params参数映射/转换成props传递给路由组件对象

实现: props: (route)=>({keyword1:route.params.keyword, keyword2: route.query.keyword })

### 解决路由跳转时的NavigationDuplicated错误

1. 问题:

编程式路由跳转到当前路径且参数没有变化时会抛出 NavigationDuplicated 错误

1. 原因分析:

vue-router3.1.0之后, 引入了push()的promise的语法, 如果没有通过参数指定回调函数就返回一个promise来指定成功/失败的回调, 且内部会判断如果要跳转的路径和参数都没有变化, 会抛出一个失败的promise

1. 解决:

方案1: 在进行跳转时, 指定跳转成功的回调函数或catch错误

|  |
| --- |
| *// catch()处理错误* **this**.$router.push(**`/search/**${**this**.**keyword**}**`**).catch(() => {}) *// 指定成功的回调函数* **this**.$router.push(**`/search/**${**this**.**keyword**}**`**, () => {})  *// 指定失败的回调函数* **this**.$router.push(**`/search/**${**this**.**keyword**}**`**, undefined, () => {}) |

方案2: 修正Vue原型上的push和replace方法

|  |
| --- |
| *// 缓存原型上的push函数* **const** originPush = VueRouter.**prototype**.push **const** originReplace = VueRouter.**prototype**.replace *// 给原型对象上的push指定新函数函数* VueRouter.**prototype**.push = **function** (location, onComplete, onAbort) {  *// 判断如果没有指定回调函数, 通过call调用源函数并使用catch来处理错误* **if** (onComplete===**undefined** && onAbort===**undefined**) {  **return** originPush.call(**this**, location, onComplete, onAbort).catch(() => {})  } **else** { *// 如果有指定任意回调函数, 通过call调用源push函数处理* originPush.call(**this**, location, onComplete, onAbort)  } } VueRouter.**prototype**.replace = **function** (location, onComplete, onAbort) {  **if** (onComplete===**undefined** && onAbort===**undefined**) {  **return** originReplace.call(**this**, location, onComplete, onAbort).catch(() => {})  } **else** {  originReplace.call(**this**, location, onComplete, onAbort)  } } |

## 引入less预编译器

### 下载依赖包

**npm install -D less less-loader**

### 组件中使用less

|  |
| --- |
| <style lang="less" scoped>  </style> |

## Footer组件

### 效果



### Footer组件编码

|  |
| --- |
| <template>    <div class="footer">      <div class="footer-container">        <div class="footerList">          <div class="footerItem">            <h4>购物指南</h4>            <ul class="footerItemCon">              <li>购物流程</li>              <li>会员介绍</li>              <li>生活旅行/团购</li>              <li>常见问题</li>              <li>购物指南</li>            </ul>          </div>          <div class="footerItem">            <h4>配送方式</h4>            <ul class="footerItemCon">              <li>上门自提</li>              <li>211限时达</li>              <li>配送服务查询</li>              <li>配送费收取标准</li>              <li>海外配送</li>            </ul>          </div>          <div class="footerItem">            <h4>支付方式</h4>            <ul class="footerItemCon">              <li>货到付款</li>              <li>在线支付</li>              <li>分期付款</li>              <li>邮局汇款</li>              <li>公司转账</li>            </ul>          </div>          <div class="footerItem">            <h4>售后服务</h4>            <ul class="footerItemCon">              <li>售后政策</li>              <li>价格保护</li>              <li>退款说明</li>              <li>返修/退换货</li>              <li>取消订单</li>            </ul>          </div>          <div class="footerItem">            <h4>特色服务</h4>            <ul class="footerItemCon">              <li>夺宝岛</li>              <li>DIY装机</li>              <li>延保服务</li>              <li>尚品汇E卡</li>              <li>尚品汇通信</li>            </ul>          </div>          <div class="footerItem">            <h4>帮助中心</h4>            <img src="./images/wx\_cz.jpg">          </div>        </div>        <div class="copyright">          <ul class="helpLink">            <li>关于我们              <span class="space"></span>            </li>            <li>联系我们              <span class="space"></span>            </li>            <li>关于我们              <span class="space"></span>            </li>            <li>商家入驻              <span class="space"></span>            </li>            <li>营销中心              <span class="space"></span>            </li>            <li>友情链接              <span class="space"></span>            </li>            <li>关于我们              <span class="space"></span>            </li>            <li>营销中心              <span class="space"></span>            </li>            <li>友情链接              <span class="space"></span>            </li>            <li>关于我们</li>          </ul>          <p>地址：北京市昌平区宏福科技园综合楼6层</p>          <p>京ICP备19006430号</p>        </div>      </div>    </div>  </template>  <script>    export default {      name: 'Footer',    }  </script>  <style lang="less" scoped>    .footer {      background-color: #eaeaea;      .footer-container {        width: 1200px;        margin: 0 auto;        padding: 0 15px;        .footerList {          padding: 20px;          border-bottom: 1px solid #e4e1e1;          border-top: 1px solid #e4e1e1;          overflow: hidden;          padding-left: 40px;          .footerItem {            width: 16.6666667%;            float: left;            h4 {              font-size: 14px;            }            .footerItemCon {              li {                line-height: 18px;              }            }            &:last-child img {              width: 121px;            }          }        }        .copyright {          padding: 20px;          .helpLink {            text-align: center;            li {              display: inline;              .space {                border-left: 1px solid #666;                width: 1px;                height: 13px;                background: #666;                margin: 8px 10px;              }            }          }          p {            margin: 10px 0;            text-align: center;          }        }      }    }  </style> |

### 控制Footer界面的显示/隐藏

解决方案: 利用路由的meta配置和v-show

1. router/routes.js

|  |
| --- |
| {    path: '/register',    component: Register,    meta: { **// 需要隐藏footer的路由添加此配置**      isHideFooter: true    }  },  {    path: '/login',    component: Login,    meta: {      isHideFooter: true    }  }, |

1. App.vue

|  |
| --- |
| <Footer v-show="!$route.meta.isHideFooter"/> |

## Home路由组件

### Home静态组件

抽取出完整的Home静态路由组件

### 从Home组件中抽取子组件(静态)

1. TypeNav: 3级分类导航
2. ListContainer: 包含轮播列表的容器
3. TodayRecommend: 今日推荐
4. Rank: 排行
5. Like: 猜你喜欢
6. Floor: 楼层
7. Brand: 品牌



## 后台应用

### 说明

1. 咱们的项目是一个前后台分离的项目: 前台应用与后台应用
2. 后台应用负责处理前台应用提交的请求, 并给前台应用返回json数据
3. 前台应用负责展现数据, 与用户交互, 与后台应用交互

### API接口文档

查看 “api文档.doc”

### 使用postman工具测试接口

1. postman是用来测试API接口的工具
2. postman也是一个活接口文档
3. 使用步骤
   1. 启动 ===> 选择登陆==> cancel ===> 进入主界面
   2. 输入url/参数进行请求测试
   3. 注意post请求体参数需要指定为json格式
   4. 保存测试接口 ==> 后面可以反复使用

## 前后台交互ajax

### 下载依赖包

**npm install -S axios nprogress**

### 封装ajax请求模块

1. api/ajax.js

|  |
| --- |
| /\*  对axios进行二次包装  1. 配置通用的基础路径和超时  2. 显示请求进度条  3. 成功返回的数据不再是response, 而直接是响应体数据response.data  4. 统一处理请求错误, 具体请求也可以选择处理或不处理  \*/  import axios from 'axios'  import NProgress from 'nprogress'  import 'nprogress/nprogress.css'  // 配置不显示右上角的旋转进度条, 只显示水平进度条  NProgress.configure({ showSpinner: false })  const service = axios.create({    baseURL: "/api", // 基础路径    timeout: 15000   // 连接请求超时时间  })  service.interceptors.request.use((config) => {    // 显示请求中的水平进度条    NProgress.start()    // 必须返回配置对象    return config  })  service.interceptors.response.use((response) => {    // 隐藏进度条    NProgress.done()    // 返回响应体数据    return response.data  }, (error) => {    // 隐藏进度条    NProgress.done()    // 统一处理一下错误    alert( `请求出错: ${error.message||'未知错误'}`)    // 后面可以选择不处理或处理    return Promise.reject(error)  })  export default service |

1. api/index.js

|  |
| --- |
| /\*  包含所有接口请求函数的模块  \*/  import ajax from './ajax'  //获取商品的三级分类列表  export const reqBaseCategoryList = ()=>ajax.get('/product/getBaseCategoryList') |

1. App.vue

|  |
| --- |
| import {reqBaseCategoryList} from '@/api'  async mounted () {      const result = await reqBaseCategoryList()      console.log('result', result)  }, |

### 配置代理

vue.config.js

|  |
| --- |
| devServer: {    proxy: {      '/api': { // 只对请求路由以/api开头的请求进行代理转发        target: 'http://182.92.128.115', // 转发的目标url        changeOrigin: true // 支持跨域      }    }  }, |

## 使用vuex管理状态

### 下载依赖包

**npm install -S vuex**

### store/modules/home.js

|  |
| --- |
| /\*  vuex管理的home模块  \*/  import {reqBaseCategoryList} from '@/api'  const state = {    baseCategoryList: [], // 所有分类的数组  }  const mutations = {    /\*    接收保存分类列表    \*/    RECEIVE\_BASE\_CATEGORY\_LIST(state, list) {      state.baseCategoryList = list    }  }  const actions = {    /\*    异步获取商品三级分类列表    \*/    async getBaseCategoryList({ commit }) {      const result = await reqBaseCategoryList();      if (result.code === 200) {        commit('RECEIVE\_BASE\_CATEGORY\_LIST', result.data)      }    },  }  const getters = {  }  export default {    state,    actions,    mutations,    getters  } |

### store/modules/index.js

|  |
| --- |
| /\*  收集所有vuex管理的模块  \*/  import home from './home'  // 向外暴露包含所有vuex管理的模块的对象  export default{    home,  } |

### store/index.js

|  |
| --- |
| /\*  Vuex最核心的管理对象store  \*/  import Vue from 'vue'  import Vuex from 'vuex'  import modules from './modules'  // 声明使用vuex插件  Vue.use(Vuex)  // 向外暴露store对象  export default new Vuex.Store({    modules, // 配置store的所有模块  }) |

### 注册store

main.js

|  |
| --- |
| **import** store **from './store'** **new** Vue({  store, // 注册vuex的store对象  ==> 所有组件对象都有一个$store属性 }) |

## 异步显示分类列表: TypeNav

### 重难点说明

1. 组件与vuex交互
2. 事件控制二三级分类列表的显示与隐藏
3. 优化高频事件触发处理: 利用lodash进行函数节流处理
4. 优化减小打包文件: 对lodash库实现按需引入
5. 解决快速移出后可能显示第一个分类的子分类列表的bug
6. 优化减少组件对象数量: 使用编程式导航代替声明式导航
7. 优化事件处理效率: 利用事件委托
8. 利用标签自定义属性携带动态数据
9. 控制一级列表的显示与隐藏
10. 一级列表显示隐藏的过渡效果
11. 优化请求执行的位置, 减少请求次数
12. 合并分类query参数与搜索的关键字params参数

### 编码实现

|  |
| --- |
| <template>    <div class="type-nav">      <div class="container" @mouseenter="isShow=true" @mouseleave="hideCategorys">        <h2 class="all">全部商品分类</h2>        <nav class="nav">          <a href="###">服装城</a>          <a href="###">美妆馆</a>          <a href="###">尚品汇超市</a>          <a href="###">全球购</a>          <a href="###">闪购</a>          <a href="###">团购</a>          <a href="###">有趣</a>          <a href="###">秒杀</a>        </nav>        <div class="sort" v-if="isShow">          <div class="all-sort-list2" @click="toSearch">            <div class="item" v-for="(c1, index) in baseCategoryList" :key="c1.categoryId"              :class="{'item-on': index===currentIndex}" @mouseenter="showCategorys(index)">              <h3>                <!-- <router-link :to="{path: '/search', query:{categoryName: c1.categoryName}}">{{c1.categoryName}}</router-link> -->                <a href="javascript:"                  :data-categoryName="c1.categoryName"                  :data-category1Id="c1.categoryId">{{c1.categoryName}}</a>              </h3>              <div class="item-list clearfix">                <div class="subitem">                  <dl class="fore" v-for="(c2, index) in c1.categoryChild" :key="c2.categoryId">                    <dt>                      <!-- <router-link :to="{path: '/search', query:{categoryName: c2.categoryName}}">{{c2.categoryName}}</router-link> -->                      <a href="javascript:"                        :data-categoryName="c2.categoryName"                        :data-category2Id="c2.categoryId">{{c2.categoryName}}</a>                    </dt>                    <dd>                      <em v-for="(c3, index) in c2.categoryChild" :key="c3.categoryId">                        <!-- <router-link :to="{path: '/search', query:{categoryName: c3.categoryName}}">{{c3.categoryName}}</router-link> -->                        <a href="javascript:"                          :data-categoryName="c3.categoryName"                          :data-category3Id="c3.categoryId">{{c3.categoryName}}</a>                      </em>                    </dd>                  </dl>                </div>              </div>            </div>          </div>        </div>      </div>    </div>  </template>  <script>    import { mapState } from 'vuex'    // import \_ from 'lodash'    import throttle from 'lodash/throttle'    export default {      name: 'TypeNav',      data () {        return {          currentIndex: -1, // 当前一级分类的下标(基2/3级分类需要显示)          isShow: false, // 是否显示一级列表        }      },      mounted () {        // 如果请求的路径是根路径则显示一级分类列表        this.isShow = this.$route.path==='/'        // 异步获取所有分类列表数据        // this.$store.dispatch('getBaseCategoryList') // 在App组件中执行, 减少请求的次数      },      computed: {        // 从vuex管理的state中读取数据到当前组件        ...mapState({          // 读取home模块的所有分类数组          baseCategoryList: state => state.home.baseCategoryList        })      },      methods: {        /\*        显示当前分类的下级分类列表 (不做节流处理)        \*/        /\*        showCategorys (index) {          console.log('showCategorys', index)          this.currentIndex = index        },        \*/        /\*        显示当前分类的下级分类列表 (使用lodash做节流处理)        \*/        // showCategorys: \_.throttle(function (index) {        showCategorys: throttle(function (index) {          // console.log('showCategorys', index)          this.currentIndex = index        }, 200), // 间隔时间为200ms        /\*        隐藏显示的2/3级分类列表        \*/        hideCategorys () {          this.currentIndex = -1          // 如果当前不是是home就隐藏一级分类列表          if (this.$route.path!=='/') {            this.isShow = false          }        },        toSearch (event) {          // 得到事件源标签上的自定义属性          const { categoryname, category1id, category2id, category3id } = event.target.dataset          // console.log(categoryname)          // 如果有分类名称, 说明点击的是某个分类项          if (categoryname) {            // 准备query参数            const query = {categoryName: categoryname}            if (category1id) {              query.category1Id = category1id            } else if (category2id) {              query.category2Id = category2id            } else if (category3id) {              query.category3Id = category3id            }            // 跳转到搜索路由, 携带准备的query参数            this.$router.push({path: '/search', query})          }        }      }    }  </script> |

## Mock/模拟数据接口

### 下载依赖包

**npm install -S mockjs**

### Web应用前后端(台)分离:

1. 后台向前台提供API接口, 只负责数据的提供和计算，而完全不处理展现
2. 前台通过Http(Ajax)请求获取数据,　在浏览器端动态构建界面显示数据

### 设计JSON数据结构

1. 理解JSON数据结构
   1. 结构: 名称, 数据类型
   2. value
   3. value可以变, 但结构不能变
2. 编写模拟JSON数据:
   1. 首页广告轮播数据: src/mock/banners.json

|  |
| --- |
| [      {          "id":"1",          "imgUrl":"/images/banner1.jpg"      },      {          "id":"2",          "imgUrl":"/images/banner2.jpg"      },      {          "id":"3",          "imgUrl":"/images/banner3.jpg"      },      {          "id":"4",          "imgUrl":"/images/banner4.jpg"      }  ] |

* 1. 首页楼层数据: src/mock/floors.json

|  |
| --- |
| [{          "id": "001",          "name": "家用电器",          "keywords": ["节能补贴", "4K电视", "空气净化器", "IH电饭煲", "滚筒洗衣机", "电热水器"],          "imgUrl": "/images/floor-1-1.png",          "navList": [{                  "url": "#",                  "text": "热门"              },              {                  "url": "#",                  "text": "大家电"              },              {                  "url": "#",                  "text": "生活电器"              },              {                  "url": "#",                  "text": "厨房电器"              },              {                  "url": "#",                  "text": "应季电器"              },              {                  "url": "#",                  "text": "空气/净水"              },              {                  "url": "#",                  "text": "高端电器"              }          ],          "carouselList": [{                  "id": "0011",                  "imgUrl": "/images/floor-1-b01.png"              },              {                  "id": "0012",                  "imgUrl": "/images/floor-1-b02.png"              },              {                  "id": "0013",                  "imgUrl": "/images/floor-1-b03.png"              }          ],          "recommendList": [              "/images/floor-1-2.png",              "/images/floor-1-3.png",              "/images/floor-1-5.png",              "/images/floor-1-6.png"          ],          "bigImg": "/images/floor-1-4.png"      },      {          "id": "002",          "name": "手机通讯",          "keywords": ["节能补贴2", "4K电视2", "空气净化器2", "IH电饭煲2", "滚筒洗衣机2", "电热水器2"],          "imgUrl": "/images/floor-1-1.png",          "navList": [{                  "url": "#",                  "text": "热门2"              },              {                  "url": "#",                  "text": "大家电2"              },              {                  "url": "#",                  "text": "生活电器2"              },              {                  "url": "#",                  "text": "厨房电器2"              },              {                  "url": "#",                  "text": "应季电器2"              },              {                  "url": "#",                  "text": "空气/净水2"              },              {                  "url": "#",                  "text": "高端电器2"              }          ],          "carouselList": [{                  "id": "0011",                  "imgUrl": "/images/floor-1-b01.png"              },              {                  "id": "0012",                  "imgUrl": "/images/floor-1-b02.png"              },              {                  "id": "0013",                  "imgUrl": "/images/floor-1-b03.png"              }          ],          "recommendList": [              "/images/floor-1-2.png",              "/images/floor-1-3.png",              "/images/floor-1-5.png",              "/images/floor-1-6.png"          ],          "bigImg": "/images/floor-1-4.png"      }  ] |

### 利用mockjs提供模拟数据

1. Mockjs: 用来拦截ajax请求, 生成随机数据返回
2. 学习
   1. <http://mockjs.com/>
   2. <https://github.com/nuysoft/Mock>
3. 使用(mock/mockServer.js)

|  |
| --- |
| /\*  利用mockjs来mock数据接口  \*/  import Mock from 'mockjs'  import banners from './banners.json'  import floors from './floors.json'  // 提供广告位轮播数据的接口  Mock.mock('/mock/banners',{      code: 200,      data: banners  })  // 提供所有楼层数据的接口  Mock.mock('/mock/floors',{      code: 200,      data: floors  }) |

### api/ajaxMock.js

|  |
| --- |
| /\*  专门请求mock接口的axios封装  \*/  import axios from 'axios'  const mockAjax = axios.create({    baseURL: "/mock", // 路径前缀    timeout: 10000 // 请求超时时间  })  mockAjax.interceptors.request.use((config) => {    return config  })  mockAjax.interceptors.response.use((response) => {    return response.data  }, (error) => {    return Promise.reject(error)  })  export default mockAjax |

### api/index.js

|  |
| --- |
| import mockAjax from './mockAjax'  // 获取广告轮播列表  export const reqBanners = ()=> mockAjax.get('/banners')  // 获取首页楼层列表  export const reqFloors = ()=> mockAjax.get('/floors') |

### vuex

store/modules/home.js

|  |
| --- |
| import { reqBaseCategoryList, reqBanners, reqFloors } from '@/api'  const state = {    baseCategoryList: [], // 所有分类的数组    banners: [], // 广告位轮播数据的数组    floors: [], // 所有楼层数据的数组  }  const mutations = {    /\*    接收分类列表    \*/    RECEIVE\_BASE\_CATEGORY\_LIST(state, list) {      state.baseCategoryList = list.slice(0, list.length-2)    },    /\*    接收广告轮播列表    \*/    RECEIVE\_BANNERS(state, banners) {      state.banners = banners    },    /\*    接收楼层列表    \*/    RECEIVE\_FLOORS(state, floors) {      state.floors = floors    },  }  const actions = {    //异步获取首页所有分类    async getBaseCategoryList({ commit }) {      const result = await reqBaseCategoryList();      if (result.code === 200) {        commit('RECEIVE\_BASE\_CATEGORY\_LIST', result.data)      }    },    //异步获取广告位轮播数据    async getBanners({ commit }) {      const result = await reqBanners();      if (result.code === 200) {        commit('RECEIVE\_BANNERS', result.data)      }    },    //异步获取所有楼层数据    async getFloors({ commit }) {      const result = await reqFloors();      if (result.code === 200) {        commit('RECEIVE\_FLOORS', result.data)      }    },  }  const getters = {  }  export default {    state,    actions,    mutations,    getters  } |

## 利用Mock接口实现动态的Home

### 重难点说明

1. 使用swiper实现静态页面轮播
2. 解决多个swiper冲突的问题
3. 解决swiper动态页面轮播的bug
4. 定义可复用的轮播组件
5. 解决Floor组件中轮播有问题的bug

### 下载依赖包

**npm install -S swiper**

### 通用的轮播组件components/Carousel

注意需要在main.js中进行全局注册

|  |
| --- |
| <**template**>  <**div class="swiper-container" ref="swiper"**>  <**div class="swiper-wrapper"**>  <**div class="swiper-slide" v-for="*item*** in **carouselList" :key="*item***.**id"**>  <**img :src="*item***.**imgUrl" alt=""**>  </**div**>  </**div**>  *<!-- 如果需要分页器 -->* <**div class="swiper-pagination"**></**div**>   *<!-- 如果需要导航按钮 -->* <**div class="swiper-button-prev"**></**div**>  <**div class="swiper-button-next"**></**div**>  </**div**> </**template**>  <**script**> **import** Vue **from 'vue' import** Swiper **from 'swiper'** *// import 'swiper/css/swiper.min.css' // 在入口js中引入*  **export default** {  **name**: **'Carousel'**,   **props**: {  **carouselList**: Array *// 外部传入轮播列表数据* },   *// 在页面初始显示后立即执行* mounted () {  ***console***.log(**'mounted'**, **this**.**carouselList**.**length**) *// 为0, banner数据是异步获取的  /\* setTimeout(() => {  // 要求: 创建swiper对象必须要在轮播列表页面显示之后执行才可以 ==> 否则轮播效果有问题  // new Swiper('.swiper-container', { // 有问题, 会匹配所有此类名元素  new Swiper(this.$refs.swiper, { // 可以, 只会匹配, 当前组件中的对应元素  // direction: 'vertical', // 垂直切换选项 默认是水平轮播  loop: true, // 循环模式   // 分页器  pagination: {  el: '.swiper-pagination',  },   // 前进后退按钮  navigation: {  nextEl: '.swiper-button-next',  prevEl: '.swiper-button-prev',  }  })  }, 1000) \*/* },   **watch**: {  *// 监视carouselList 一般监视就可以* **carouselList**: {  handler () { *// 说明carouselList状态数据发了改变, 但界面还没有更新  // 只有数组中有数据, 才需要创建swiper对象* **if** (**this**.**carouselList**.**length**===0) **return   *console***.log(**'watch carouselList'**, **this**.**carouselList**.**length**) *// 执行3次行  /\*   数据绑定流程 ==> 更新状态数据 ==> 同步调用监视的回调函数 ==> 界面就会自动 `异步`更新  \*/  // vm.$nextTick( [callback] )  // 将回调延迟到下次 DOM 更新循环(更新界面)之后执行。$nextTick()在修改数据之后立即调用，然后等待 DOM 更新  // this.$nextTick(() => {// 回调函数在界面更新之后执行* Vue.nextTick(() => {*// 回调函数在界面更新之后执行  // 必须在轮播列表界面显示之后创建* **new** Swiper(**this**.$refs.**swiper**, { *// 可以, 只会匹配, 当前组件中的对应元素  // direction: 'vertical', // 垂直切换选项 默认是水平轮播* **loop**: **true**, *// 循环模式   // 分页器* **pagination**: {  **el**: **'.swiper-pagination'**,  },   *// 前进后退按钮* **navigation**: {  **nextEl**: **'.swiper-button-next'**,  **prevEl**: **'.swiper-button-prev'**,  },  })  })  },  **immediate**: **true**, *// 在初始显示时就立即执行一次, 默认是false(只有数据改变才立即执行)* },  }, } </**script**> |

### 首页的ListContainer组件

|  |
| --- |
| <**template**>  <**div class="list-container"**>  <**div class="sortList clearfix"**>  <**div class="center"**>  *<!--banner轮播-->  <!--   banners为空数组 ==> 渲染Carousel ===> 创建Carousel, 调用mounted, 没有数据(长度为0)  异步获取banners数组 ==> 更新渲染Carousel ===> 调用watch的回调函数  -->* <**Carousel :carouselList="banners"**/>  </**div**>  <**div class="right"**>  <**div class="news"**>  <**h4**>  <**em class="fl"**>谷粒商城快报</**em**>  <**span class="fr tip"**>更多 ></**span**>  </**h4**>  <**div class="clearix"**></**div**>  <**ul class="news-list unstyled"**>  <**li**>  <**span class="bold"**>[特惠]</**span**>备战开学季 全民半价购数码  </**li**>  <**li**>  <**span class="bold"**>[公告]</**span**>备战开学季 全民半价购数码  </**li**>  <**li**>  <**span class="bold"**>[特惠]</**span**>备战开学季 全民半价购数码  </**li**>  <**li**>  <**span class="bold"**>[公告]</**span**>备战开学季 全民半价购数码  </**li**>  <**li**>  <**span class="bold"**>[特惠]</**span**>备战开学季 全民半价购数码  </**li**>  </**ul**>  </**div**>  <**ul class="lifeservices"**>  <**li class=" life-item "**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>话费</**span**>  </**li**>  <**li class=" life-item "**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>机票</**span**>  </**li**>  <**li class=" life-item "**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>电影票</**span**>  </**li**>  <**li class=" life-item "**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>游戏</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>彩票</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>加油站</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>酒店</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>火车票</**span**>  </**li**>  <**li class=" life-item "**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>众筹</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>理财</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>礼品卡</**span**>  </**li**>  <**li class=" life-item"**>  <**i class="list-item"**></**i**>  <**span class="service-intro"**>白条</**span**>  </**li**>  </**ul**>  <**div class="ads"**>  <**img src="./images/ad1.png"** />  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **import** { mapState } **from 'vuex'   export default** {  **name**: **'ListContainer'**,   **computed**: {  ...mapState({  banners: state => state.**home**.**banners** })  }  } </**script**> |

### 首页的Floor组件

|  |
| --- |
| <**template**>  <**div class="floor"**>  <**div class="py-container"**>  <**div class="title clearfix"**>  <**h3 class="fl"**>{{floor.**name**}}</**h3**>  <**div class="fr"**>  <**ul class="nav-tabs clearfix"**>  <**li class="active" v-for="**(nav, ***index***) in floor.navList**" :key="**nav.**text"**>  <**a href="#tab1" data-toggle="tab"**>{{nav.**text**}}</**a**>  </**li**>  </**ul**>  </**div**>  </**div**>  <**div class="tab-content"**>  <**div class="tab-pane"**>  <**div class="floor-1"**>  <**div class="blockgary"**>  <**ul class="jd-list"**>  <**li v-for="keyword** in floor.keywords**" :key="keyword"**>{{**keyword**}}</**li**>  </**ul**>  <**img :src="**floor.**imgUrl"** />  </**div**>  <**div class="floorBanner"**>  <**Carousel :carouselList="**floor.**carouselList"**/>  </**div**>  <**div class="split"**>  <**span class="floor-x-line"**></**span**>  <**div class="floor-conver-pit"**>  <**img :src="**floor.recommendList[0]**"** />  </**div**>  <**div class="floor-conver-pit"**>  <**img :src="**floor.recommendList[1]**"** />  </**div**>  </**div**>  <**div class="split center"**>  <**img :src="**floor.bigImg**"** />  </**div**>  <**div class="split"**>  <**span class="floor-x-line"**></**span**>  <**div class="floor-conver-pit"**>  <**img :src="**floor.recommendList[2]**"** />  </**div**>  <**div class="floor-conver-pit"**>  <**img :src="**floor.recommendList[3]**"** />  </**div**>  </**div**>  </**div**>  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**> **export default** {  **name**: **'Floor'**,  **props**: {  **floor**: ***Object*** } } </**script**> |

### 在Home组件中使用

|  |
| --- |
| <!--列表-->  <**ListContainer** />  <!--楼层-->  <**Floor v-for="*floor*** in **floors" :key="*floor***.**id" :floor="*floor*"**/>  mounted () {  *// 触发vuex的异步action调用, 从mock接口请求数据到state中* **this**.$store.**dispatch**(**'getBanners'**)  **this**.$store.**dispatch**(**'getFloors'**) },  **computed**: {  ...mapState({  floors: state => state.**home**.**floors** }) }, |

## Search路由

### 重难点说明

1. 搜索查询条件参数理解与准备
2. 组件动态数据显示
3. 根据分类和关键字进行搜索
4. 根据品牌进行搜索
5. 根据属性进行搜索
6. 排序搜索
7. 自定义分页组件

### 接口请求函数

api/index.js

|  |
| --- |
| *// 请求搜索匹配的商品相关数据* **export const** *reqProductList* = (searchParams) => ajax.post(**'/list'**, searchParams) |

### vuex管理的搜索模块

store/modules/search.js

|  |
| --- |
| */\*  管理搜索模块的数据 \*/* **import** {reqProductList} **from '@/api'  const** state = {  **productList**: {}, *// 搜索出的商品列表相关数据的对象* } **const** mutations = {  */\*   接收保存商品列表相关数据对象  \*/* RECEIVE\_PRODUCT\_LIST (state, productList) {  state.**productList** = productList  } } **const** actions = {   */\*   根据指定的搜索条件, 异步获取商品列表的action  \*/* **async** getProductList ({commit}, searchParams) {  *// 如果不想改变组件中的options* searchParams = {...searchParams}   *// 过滤掉searchParams对象中所有属性值为空串的属性  // Object.keys(obj): 得到对象本身所有属性名的数组* Object.keys(searchParams).forEach(key => {  **if** (searchParams[key]===**''**) {  **delete** searchParams[key] *// 组件中的options也改变了* }  })     *// 1. ajax请求, 获取数据* **const** result = **await** reqProductList(searchParams)  *// 2. 如果成功, 提交给mutation* **if** (result.**code**===200) {  **const** productList = result.**data** commit(**'RECEIVE\_PRODUCT\_LIST'**, productList)  }  } } **const** getters = {  *// 返回品牌列表* trademarkList (state) {  **return** state.**productList**.trademarkList || []  },   *// 返回属性列表* attrsList (state) {  **return** state.**productList**.**attrsList** || []  },   *// 商品列表* goodsList (state) {  **return** state.**productList**.goodsList || []  } }  **export default** {  state,  mutations,  actions,  getters } |

### Search的子组件: 属性选择器

pages/Search/SearchSelector/SearchSelector.vue

|  |
| --- |
| <**template**>  <**div class="clearfix selector"**>  <**div class="type-wrap logo"**>  <**div class="fl key brand"**>品牌</**div**>  <**div class="value logos"**>  <**ul class="logo-list"**>  <**li v-for="**tm in trademarkList**" :key="**tm.tmId**"   @click="setTrademark**(**`**${tm.tmId}**:**${tm.tmName}**`**)**"**>{{tm.tmName}}</**li**>  </**ul**>  </**div**>  <**div class="ext"**>  <**a href="javascript:void(0);" class="sui-btn"**>多选</**a**>  <**a href="javascript:void(0);"**>更多</**a**>  </**div**>  </**div**>  <**div class="type-wrap" v-for="attr** in attrList**" :key="attr**.attrId**"**>  <**div class="fl key"**>{{**attr**.attrName}}</**div**>  <**div class="fl value"**>  <**ul class="type-list"**>  <**li v-for="*value*** in **attr**.attrValueList**" :key="*value*"**>  <**a href="javascript:void(0);" @click="***addProp*(**attr**.attrId, ***value***, **attr**.attrName)**"**>{{***value***}}</**a**>  </**li**>  </**ul**>  </**div**>  <**div class="fl ext"**></**div**>  </**div**>    </**div**> </**template**>  <**script**>  **import** {mapState} **from 'vuex'  export default** {  **name**: **'SearchSelector'**,   **props**: {  **setTrademark**: Function, *// 是用于更新父组件的trademark数据的函数* **addProp**: Function, *// 是用于更新父组件的props数据的函数* },   **computed**: {  ...mapState({  trademarkList: state => state.**search**.**productList**.trademarkList, *// 品牌列表* attrList: state => state.**search**.**productList**.attrsList, *// 属性列表* })  }  } </**script**>  <**style lang="less" scoped**>  .**selector** {  **border**: 1**px solid #ddd**;  **margin-bottom**: 5**px**;  **overflow**: **hidden**;   .**logo** {  **border-top**: 0;  **margin**: 0;  **position**: **relative**;  **overflow**: **hidden**;   .**key** {  **padding-bottom**: 87**px !important**;  }  }   .**type-wrap** {  **margin**: 0;  **position**: **relative**;  **border-top**: 1**px solid #ddd**;  **overflow**: **hidden**;   .**key** {  **width**: 100**px**;  **background**: **#f1f1f1**;  **line-height**: 26**px**;  **text-align**: **right**;  **padding**: 10**px** 10**px** 0 15**px**;  **float**: **left**;  }   .**value** {  **overflow**: **hidden**;  **padding**: 10**px** 0 0 15**px**;  **color**: **#333**;  **margin-left**: 120**px**;  **padding-right**: 90**px**;   .**logo-list** {  **li** {  **float**: **left**;  **border**: 1**px solid #e4e4e4**;  **margin**: -1**px** -1**px** 0 0;  **width**: 105**px**;  **height**: 52**px**;  **text-align**: **center**;  **line-height**: 52**px**;  **overflow**: **hidden**;  **text-overflow**: **ellipsis**;  **white-space**: **nowrap**;  **font-weight**: 700;  **color**: **#e1251b**;  **font-style**: **italic**;  **font-size**: 14**px**;   **img** {  **max-width**: 100%;  **vertical-align**: **middle**;  }  }  }   .**type-list** {  **li** {  **float**: **left**;  **display**: **block**;  **margin-right**: 30**px**;  **line-height**: 26**px**;   **a** {  **text-decoration**: **none**;  **color**: **#666**;  }  }  }  }   .**ext** {  **position**: **absolute**;  **top**: 10**px**;  **right**: 10**px**;   .**sui-btn** {  **display**: **inline-block**;  **padding**: 2**px** 14**px**;  **box-sizing**: **border-box**;  **margin-bottom**: 0;  **font-size**: 12**px**;  **line-height**: 18**px**;  **text-align**: **center**;  **vertical-align**: **middle**;  **cursor**: **pointer**;  **padding**: 0 10**px**;  **background**: **#fff**;  **border**: 1**px solid #d5d5d5**;  }   **a** {  **color**: **#666**;  }  }  }  } </**style**> |

### 通用的分页组件Pagination

1. 静态组件

|  |
| --- |
| <**template**>  <**div class="pagination"**>  <**button**>1</**button**>  <**button**>上一页</**button**>  <**button**>···</**button**>   <**button**>3</**button**>  <**button**>4</**button**>  <**button**>5</**button**>  <**button**>6</**button**>  <**button**>7</**button**>    <**button**>···</**button**>  <**button**>9</**button**>  <**button**>上一页</**button**>    <**button style="margin-left: 30px"**>共 60 条</**button**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **"Pagination"**,  } </**script**>  <**style lang="less" scoped**>  .**pagination** {  **button** {  **margin**: 0 5**px**;  **background-color**: **#f4f4f5**;  **color**: **#606266**;  **outline**: **none**;  **border-radius**: 2**px**;  **padding**: 0 4**px**;  **vertical-align**: **top**;  **display**: **inline-block**;  **font-size**: 13**px**;  **min-width**: 35.5**px**;  **height**: 28**px**;  **line-height**: 28**px**;  **cursor**: **pointer**;  **box-sizing**: **border-box**;  **text-align**: **center**;  **border**: 0;   **&**[**disabled**] {  **color**: **#c0c4cc**;  **cursor**: **not-allowed**;  }   **&**.**active** {  **cursor**: **not-allowed**;  **background-color**: **#409eff**;  **color**: **#fff**;  }  }  } </**style**> |

1. 动态组件

|  |
| --- |
| <**template**>  <**div class="pagination"**>  *<!-- 当前页码等于1就不可操作 -->* <**button :disabled="mcPage**===1**" @click="**changeCurrentPage(**mcPage**-1)**"**>上一页</**button**>  *<!-- 只有start大于1 -->* <**button v-if="**startEnd.**start**>1**" @click="**changeCurrentPage(1)**"**>1</**button**>  *<!-- 只有start大于2 -->* <**button disabled v-if="**startEnd.**start**>2**"**>···</**button**>   *<!-- 连续页码 -->* <**button v-for="*item*** in startEnd.**end" v-if="*item***>=startEnd.**start" :key="*item*"  @click="**changeCurrentPage(***item***)**" :class="**{**active**: **mcPage**===***item***}**"**>  {{***item***}}  </**button**>  *<!-- 只有end<totalPages-1才显示 -->* <**button disabled v-if="**startEnd.**end**<totalPages-1**"**>···</**button**>  *<!-- 只有end<totalPages才显示 -->* <**button v-if="**startEnd.**end**<totalPages**" @click="**changeCurrentPage(totalPages)**"**>{{totalPages}}</**button**>  *<!-- 当前页码等于总页码就不可操作 -->* <**button :disabled="mcPage**===totalPages**" @click="**changeCurrentPage(**mcPage**+1)**"**>下一页</**button**>  *<!-- 总记录数 -->* <**button style="margin-left: 30px"**>共 {{total}} 条</**button**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **"Pagination"**,   **props**: {  **currentPage**: { *// 当前页码* **type**: ***Number***,  **default**: 1  },  **pageSize**: { *// 每页数量* **type**: ***Number***,  **default**: 5  },  **total**: { *// 总数量* **type**: ***Number***,  **default**: 0  },  **showPageNo**: { *// 连续页码数* **type**: ***Number***,  **default**: 5  }  },   data () {  **return** {  **mcPage**: **this**.**currentPage** *// 保存自己的当前页码* }  },   **computed**: {   */\*   总页码数  依赖数据:   总数量: total   每页数量: pageSize   \*/* totalPages () {  *// 取出依赖数据 31 5 ==> 7* **const** {total, pageSize} = **this** *// 返回计算后的结果* **return *Math***.ceil(total/pageSize)  },   */\*   返回连续页码的开始页码(start)与结束页码(end):   比如: {start: 3, end: 7}  依赖数据:  当前页码: mcPage  最大连续页码数: showPageNo  总页码数: totalPages  注意:  start的最小值为1  end的最大值为totalPages  start与end之间的最大差值为showPageNo-1  \*/* startEnd () {  **const** {mcPage, showPageNo, totalPages} = **this** *// 计算start  /\*   mcPage showPageNo totalPages start到end  4 5 10 23[4]56  \*/* **let** start = mcPage - ***Math***.floor(showPageNo/2) *// 4 - 2  /\*   mcPage showPageNo totalPages start到end  2 5 10 1[2]345  但start上面计算得到是: 0  \*/  // start的最小值是1, 如果小于1, 修正为1* **if** (start<1) {  start = 1  }   *// 计算end  /\*   mcPage showPageNo totalPages start到end  4 5 10 23[4]56  \*/  // start与end之间的最大差值为showPageNo-1* **let** end = start + showPageNo - 1 *// 2 + 5 -1    /\*   mcPage showPageNo totalPages start到end  4 5 5 123[4]5  但上面计算的end为6, 应该为5 ==> end = totalPages  start为2, 应该为1 ==> start = end - showPageNo + 1  \*/  // 如果end超过了totalPages, 修正为totalPages* **if** (end > totalPages) {  end = totalPages  *// 根据最大连续页码修正start* start = end - showPageNo + 1   */\*   mcPage showPageNo totalPages start到end  4 5 4 123[4]  上面计算  start为0 应该为1  end为4 没问题  \*/  // start不能小于最小值1* **if** (start<1) {  start = 1  }  }   **return** {start, end}  }  },   **watch**: {  */\*  当接收的currentPage发生改变时调用   \*/* currentPage (value) {  *// 将当前页码指定为外部传入的值* **this**.**mcPage** = value  }  },   **methods**: {   */\*   将当前页码改为指定页码  \*/* changeCurrentPage (page) {  *// 修改当前页码* **this**.**mcPage** = page  *// 通知外部父组件* **this**.$emit(**'currentChange'**, page)  }  },  } </**script**>  <**style lang="less" scoped**>  .**pagination** {  **button** {  **margin**: 0 5**px**;  **background-color**: **#f4f4f5**;  **color**: **#606266**;  **outline**: **none**;  **border-radius**: 2**px**;  **padding**: 0 4**px**;  **vertical-align**: **top**;  **display**: **inline-block**;  **font-size**: 13**px**;  **min-width**: 35.5**px**;  **height**: 28**px**;  **line-height**: 28**px**;  **cursor**: **pointer**;  **box-sizing**: **border-box**;  **text-align**: **center**;  **border**: 0;   **&**[**disabled**] {  **color**: **#c0c4cc**;  **cursor**: **not-allowed**;  }   **&**.**active** {  **cursor**: **not-allowed**;  **background-color**: **#409eff**;  **color**: **#fff**;  }  }  } </**style**> |

1. 注册为全局组件

|  |
| --- |
| import Pagination from './components/Pagination'  Vue.component('Pagination', Pagination) // 全局使用<Pagination/> <pagination/> |

### TypeNav组件

|  |
| --- |
| */\*  点击某个分类项跳转到search路由 \*/* toSearch(event) { *// 只绑定的一个点击监听  // console.dir(event.target)  // 得到所有标签上的data自定义属性* **const** dataset = event.**target**.dataset  *// console.log('dataset', dataset)  // 取出自定义属性值* **const** {  categoryname,  category1id,  category2id,  category3id  } = dataset   *//if (event.target.tagName==='A') { // 如果点击的是a标签就可以跳转了* **if** (categoryname) { *// 必然点击的是分类项  // 准备query参数对象* **const** query = {  **categoryName**: categoryname  }  **if** (category1id) {  query.**category1Id** = category1id  } **else if** (category2id) {  query.**category2Id** = category2id  } **else if** (category3id) {  query.**category3Id** = category3id  }   *// 得到当前路由路径 / 或者 /search 或者 /search/xxx* **const** {  path,  params  } = **this**.$route   **this**.hideFirst()   *// 如果当前已经在搜索界面* **if** (path.indexOf(**'/search'**) === 0) {  *// 跳转到搜索, path为原本的路径(可能携带了params参数)* **this**.$router.replace({  **name**: **'search'**,  params,  query  })  } **else** { *// 当前没在搜索界面  // 跳转路由, 并携带query参数* **this**.$router.push({  **path**: **'/search'**,  query  })  }  } }, |

### Header组件

|  |
| --- |
| mounted () {  *// 通过全局总线绑定removeKeyword事件监听* **this**.**$bus**.$on(**'removeKeyword'**, () => {  **this**.**keyword** = **''** *// 置空我们的搜索关键字* }) },  toSearch () {  *// 得到当前的请求路径和query参数对象* **const** {path, query} = **this**.$route  **if** (**this**.**keyword**) {  *// 如果当前在搜索页面, 需要携带params和query参数* **if** (path.indexOf(**'/search'**)===0) {  **this**.$router.push({  **name**: **'search'**,  **params**: {**keyword**: **this**.**keyword**},  query  }) *// 可以* } **else** { *// 如果不在, 只需要携带params参数* **this**.$router.push({**name**: **'search'**, **params**: {**keyword**: **this**.**keyword**}}) *// 可以* }  } **else** {  **if** (path.indexOf(**'/search'**)===0) {  **this**.$router.push({**name**: **'search'**, query})  } **else** {  **this**.$router.push({**name**: **'search'**})  }  } } |

### Search路由组件

pages/Search/index.vue

|  |
| --- |
| <**template**>  <**div**>  <**TypeNav** />   <**div class="main"**>  <**div class="py-container"**>  *<!--bread-->* <**div class="bread"**>  <**ul class="fl sui-breadcrumb"**>  <**li**>  <**a href="#"**>全部结果</**a**>  </**li**>  </**ul**>  <**ul class="fl sui-tag"**>  <**li class="with-x" v-if="*options***.**categoryName"**>  {{***options***.**categoryName**}}  <**i @click="**removeCategory**"**>×</**i**>  </**li**>  <**li class="with-x" v-if="*options***.**keyword"**>  {{***options***.**keyword**}}  <**i @click="**removeKeyword**"**>×</**i**>  </**li**>  <**li class="with-x" v-if="*options***.**trademark"**>  {{***options***.**trademark**}}  <**i @click="**removeTrademark**"**>×</**i**>  </**li**>   <**li class="with-x" v-for="**(prop, ***index***) in ***options***.**props" :key="**prop**"**>  {{prop}}  <**i @click="**removeProp(***index***)**"**>×</**i**>  </**li**>  </**ul**>  </**div**>  *<!--selector-->* <**SearchSelector :setTrademark="setTrademark" :addProp="***addProp***"**/>  *<!--details-->* <**div class="details clearfix"**>  <**div class="sui-navbar"**>  <**div class="navbar-inner filter"**>  <**ul class="sui-nav"**>  <**li :class="**{**active**: ***options***.**order**.indexOf(**'1'**)===0}**" @click="setOrder**(**'1'**)**"**>  <**a href="javascript:"**>  综合  *<!-- 1:desc icondown 1:asc iconup 其它不显示 -->* <**i class="iconfont" :class="**orderIcon**" v-if="*options***.**order**.indexOf(**'1'**)===0**"**></**i**>  </**a**>  </**li**>  <**li**>  <**a href="javascript:"**>销量</**a**>  </**li**>  <**li**>  <**a href="javascript:"**>新品</**a**>  </**li**>  <**li**>  <**a href="javascript:"**>评价</**a**>  </**li**>  <**li :class="**{**active**: ***options***.**order**.indexOf(**'2'**)===0}**" @click="setOrder**(**'2'**)**"**>  <**a href="javascript:"**>  价格  <**i class="iconfont" :class="**orderIcon**" v-if="*options***.**order**.indexOf(**'2'**)===0**"**></**i**>  </**a**>  </**li**>  </**ul**>  </**div**>  </**div**>  <**div class="goods-list"**>  <**ul class="yui3-g"**>  <**li class="yui3-u-1-5" v-for="**goods in **productList**.goodsList**" :key="**goods.**id"**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**a href="javascript:"**>  <**img :src="**goods.defaultImg**"** />  </**a**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>{{goods.price}}</**i**>  </**strong**>  </**div**>  <**div class="attr"**>  <**a href="javascript:"**>{{goods.**title**}}</**a**>  </**div**>  <**div class="commit"**>  <**i class="command"**>已有<**span**>2000</**span**>人评价</**i**>  </**div**>  <**div class="operate"**>  <**a href="success-cart.html" target="\_blank" class="sui-btn btn-bordered btn-danger"**>加入购物车</**a**>  <**a href="javascript:void(0);" class="sui-btn btn-bordered"**>收藏</**a**>  </**div**>  </**div**>  </**li**>  </**ul**>  </**div**>    <**Pagination   :currentPage="*options***.**pageNo"   :pageSize="*options***.**pageSize"   :total="productList**.total**"  :showPageNo="**5**"  @currentChange="**getProductList**"** />  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **import** {mapState} **from 'vuex'  import** SearchSelector **from './SearchSelector/SearchSelector'  export default** {  **name**: **'Search'**,  *// props: ['keyword3', 'keyword4'],* data () {  **return** {  *// 包含所有用于搜索请求的参数数据的对象* **options**: {  **category1Id**: **''**, *// 一级分类ID* **category2Id**: **''**, *// 二级分类ID* **category3Id**: **''**, *// 三级分类ID* **categoryName**: **''**, *// 分类名称* **keyword**: **''**, *// 关键字* **trademark**: **''**, *// 品牌 "ID:品牌名称"* **props**: [], *// 商品属性的数组: ["属性ID:属性值:属性名"] 示例: ["2:6.0～6.24英寸:屏幕尺寸"]* **order**: **'2:desc'**, *// 排序方式 1: 综合,2: 价格 asc: 升序,desc: 降序 示例: "1:desc"* **pageNo**: 1, *// 当前页码* **pageSize**: 5, *// 每页数量* }  }  },   **computed**: {  ...mapState({  productList: state => state.**search**.**productList** }),   */\*   返回排序方式的图标类名  \*/* orderIcon () {  **return this**.**options**.**order**.split(**':'**)[1]===**'desc'** ? **'icondown'** : **'iconup'** }  },   **watch**: {  */\*   当路由跳转时只有路由参数发生了变化  \*/* $route () {  *// 更新options* **this**.updateOptions()  *// 请求获取数据* **this**.getProductList()  }  },   */\*   放初始同步更新data数据的代码  \*/* beforeMount () {  **this**.updateOptions()  },   */\*   初始异步更新的代码  \*/* mounted () {  ***console***.log(**'Search mounted()'**)  */\* this.$store.dispatch('getProductList', {  "category3Id": "61",  "categoryName": "手机",  "keyword": "小米",  "order": "1:desc",  "pageNo": 1,  "pageSize": 10,  "props": ["1:1700-2799:价格", "2:6.65-6.74英寸:屏幕尺寸"],  "trademark": "4:小米"  }) \*/   // this.$store.dispatch('getProductList', this.options)* **this**.getProductList()   */\*   const obj1 = {a: 1, b: 2, c: 3}  const obj2 = {b: 4, d: 5}  const obj3 = {...obj1, ...obj2, d: 6} // {a: 1, b: 4, c: 3, d: 6}  \*/* },   **methods**: {   */\*   异步获取指定页码(默认为1)的数据数据  \*/* getProductList (currentPage=1) {  **this**.**options**.**pageNo** = currentPage  **this**.$store.**dispatch**(**'getProductList'**, **this**.**options**)  },   */\*   设置新的排序方式  \*/* setOrder (newOrderFlag) {*// orderFlag为1/2* **let** [orderFlag, orderType] = **this**.**options**.**order**.split(**':'**)  **if** (newOrderFlag===orderFlag) {  orderType = orderType===**'desc'** ? **'asc'** : **'desc'** } **else** {  orderFlag = newOrderFlag  orderType = **'desc'** }  **this**.**options**.**order** = orderFlag + **':'** + orderType   **this**.getProductList()  },   */\*   删除指定下标的属性条件  \*/* removeProp (index) {  *// 删除对应的prop* **this**.**options**.**props**.splice(index, 1)  *// 重新请求数据显示* **this**.getProductList()  },   */\*   添加一个属性条件  \*/* addProp (attrId, value, attrName) {   *// 组装prop* **const** prop = **`**${attrId}**:**${value}**:**${attrName}**`** *// 如果已经添加过了当前属性, 直接结束  // ["属性ID:属性值:属性名"]* **if** (**this**.**options**.**props**.indexOf(prop)!==-1) **return** *// 向options中的props添加一个prop* **this**.**options**.**props**.push(prop)   *// 重新请求数据显示* **this**.getProductList()  },   */\*   设置新的品牌条件数据  \*/* setTrademark (trademark) {  *// 更新options中的trademark* **this**.**options**.**trademark** = trademark  *// 重新请求获取商品列表显示* **this**.getProductList()  },   */\*   移除品牌搜索条件  \*/* removeTrademark () {  *// 重置trademark数据* **this**.**options**.**trademark** = **''** *// 重新请求获取商品列表显示* **this**.getProductList()  },   */\*   移除分类的搜索条件  \*/* removeCategory () {  *// 重置分类的条件数据* **this**.**options**.categoryName = **''  this**.**options**.category1Id = **''  this**.**options**.category2Id = **''  this**.**options**.category3Id = **''** *// 重新获取数据  // this.$store.dispatch('getProductList', this.options) // 不可以  // 重新跳转到当前路由, 不再携带query参数, 只携带原本的params参数* **this**.$router.replace(**this**.$route.path) *// $route.path不带query参数, 但带params参数(如果有)* },   */\*   移除关键字的搜索条件  \*/* removeKeyword () {  *// 重置分类的条件数据* **this**.**options**.keyword = **''** *// 重新获取数据  // this.$store.dispatch('getProductList', this.options) // 不可以   // 重新跳转到当前路由, 不再携带params参数, 只携带原本的query参数* **this**.$router.replace({**name**: **'search'**, **query**: **this**.$route.query})   *// 通知Header组件也删除输入的关键字  // 在Search, 通过事件总线对象来分发事件* **this**.**$bus**.$emit(**'removeKeyword'**)  },   */\*   根据query和params来更新options数据  \*/* updateOptions () {  *// 根据query和params更新options* **const** {categoryName, category1Id, category2Id, category3Id} = **this**.$route.query  **const** {keyword} = **this**.$route.params  **this**.**options** = {  ...**this**.**options**,  categoryName,  category1Id,  category2Id,  category3Id,  keyword,  }  },   */\* handleCurrentChange (currentPage) {  this.options.pageNo = currentPage  this.$store.dispatch('getProductList', this.options)  } \*/* },   **components**: {  SearchSelector  }  } </**script**>  <**style lang="less" scoped**>  .**main** {  **margin**: 10**px** 0;   .**py-container** {  **width**: 1200**px**;  **margin**: 0 **auto**;   .**bread** {  **margin-bottom**: 5**px**;  **overflow**: **hidden**;   .**sui-breadcrumb** {  **padding**: 3**px** 15**px**;  **margin**: 0;  **font-weight**: 400;  **border-radius**: 3**px**;  **float**: **left**;   **li** {  **display**: **inline-block**;  **line-height**: 18**px**;   **a** {  **color**: **#666**;  **text-decoration**: **none**;   **&**:**hover** {  **color**: **#4cb9fc**;  }  }  }  }   .**sui-tag** {  **margin-top**: -5**px**;  **list-style**: **none**;  **font-size**: 0;  **line-height**: 0;  **padding**: 5**px** 0 0;  **margin-bottom**: 18**px**;  **float**: **left**;   .**with-x** {  **font-size**: 12**px**;  **margin**: 0 5**px** 5**px** 0;  **display**: **inline-block**;  **overflow**: **hidden**;  **color**: **#000**;  **background**: **#f7f7f7**;  **padding**: 0 7**px**;  **height**: 20**px**;  **line-height**: 20**px**;  **border**: 1**px solid #dedede**;  **white-space**: **nowrap**;  **transition**: **color** 400**ms**;  **cursor**: **pointer**;   **i** {  **margin-left**: 10**px**;  **cursor**: **pointer**;  **font**: 400 14**px tahoma**;  **display**: **inline-block**;  **height**: 100%;  **vertical-align**: **middle**;  }   **&**:**hover** {  **color**: **#28a3ef**;  }  }  }  }   .**details** {  **margin-bottom**: 5**px**;   .**sui-navbar** {  **overflow**: **visible**;  **margin-bottom**: 0;   .**filter** {  **min-height**: 40**px**;  **padding-right**: 20**px**;  **background**: **#fbfbfb**;  **border**: 1**px solid #e2e2e2**;  **padding-left**: 0;  **border-radius**: 0;  **box-shadow**: 0 1**px** 4**px rgba**(0, 0, 0, 0.065);   .**sui-nav** {  **position**: **relative**;  **left**: 0;  **display**: **block**;  **float**: **left**;  **margin**: 0 10**px** 0 0;   **li** {  **float**: **left**;  **line-height**: 18**px**;   **a** {  **display**: **block**;  **cursor**: **pointer**;  **padding**: 11**px** 15**px**;  **color**: **#777**;  **text-decoration**: **none**;  }   **&**.**active** {  **a** {  **background**: **#e1251b**;  **color**: **#fff**;  }  }  }  }  }  }   .**goods-list** {  **margin**: 20**px** 0;   **ul** {  **display**: **flex**;  **flex-wrap**: **wrap**;   **li** {  **height**: 100%;  **width**: 20%;  **margin-top**: 10**px**;  **line-height**: 28**px**;   .**list-wrap** {  .**p-img** {  **padding-left**: 15**px**;  **width**: 215**px**;  **height**: 255**px**;   **a** {  **color**: **#666**;   **img** {  **max-width**: 100%;  **height**: **auto**;  **vertical-align**: **middle**;  }  }  }   .**price** {  **padding-left**: 15**px**;  **font-size**: 18**px**;  **color**: **#c81623**;   **strong** {  **font-weight**: 700;   **i** {  **margin-left**: -5**px**;  }  }  }   .**attr** {  **padding-left**: 15**px**;  **width**: 85%;  **overflow**: **hidden**;  **margin-bottom**: 8**px**;  **min-height**: 38**px**;  **cursor**: **pointer**;  **line-height**: 1.8;  **display**: **-webkit-box**;  **-webkit-box-orient**: **vertical**;  **-webkit-line-clamp**: 2;   **a** {  **color**: **#333**;  **text-decoration**: **none**;  }  }   .**commit** {  **padding-left**: 15**px**;  **height**: 22**px**;  **font-size**: 13**px**;  **color**: **#a7a7a7**;   **span** {  **font-weight**: 700;  **color**: **#646fb0**;  }  }   .**operate** {  **padding**: 12**px** 15**px**;   .**sui-btn** {  **display**: **inline-block**;  **padding**: 2**px** 14**px**;  **box-sizing**: **border-box**;  **margin-bottom**: 0;  **font-size**: 12**px**;  **line-height**: 18**px**;  **text-align**: **center**;  **vertical-align**: **middle**;  **cursor**: **pointer**;  **border-radius**: 0;  **background-color**: **transparent**;  **margin-right**: 15**px**;  }   .**btn-bordered** {  **min-width**: 85**px**;  **background-color**: **transparent**;  **border**: 1**px solid #8c8c8c**;  **color**: **#8c8c8c**;   **&**:**hover** {  **border**: 1**px solid #666**;  **color**: **#fff !important**;  **background-color**: **#666**;  **text-decoration**: **none**;  }  }   .**btn-danger** {  **border**: 1**px solid #e1251b**;  **color**: **#e1251b**;   **&**:**hover** {  **border**: 1**px solid #e1251b**;  **background-color**: **#e1251b**;  **color**: **white !important**;  **text-decoration**: **none**;  }  }  }  }  }  }  }   .**page** {  **width**: 733**px**;  **height**: 66**px**;  **overflow**: **hidden**;  **float**: **right**;   .**sui-pagination** {  **margin**: 18**px** 0;   **ul** {  **margin-left**: 0;  **margin-bottom**: 0;  **vertical-align**: **middle**;  **width**: 490**px**;  **float**: **left**;   **li** {  **line-height**: 18**px**;  **display**: **inline-block**;   **a** {  **position**: **relative**;  **float**: **left**;  **line-height**: 18**px**;  **text-decoration**: **none**;  **background-color**: **#fff**;  **border**: 1**px solid #e0e9ee**;  **margin-left**: -1**px**;  **font-size**: 14**px**;  **padding**: 9**px** 18**px**;  **color**: **#333**;  }   **&**.**active** {  **a** {  **background-color**: **#fff**;  **color**: **#e1251b**;  **border-color**: **#fff**;  **cursor**: **default**;  }  }   **&**.**prev** {  **a** {  **background-color**: **#fafafa**;  }  }   **&**.**disabled** {  **a** {  **color**: **#999**;  **cursor**: **default**;  }  }   **&**.**dotted** {  **span** {  **margin-left**: -1**px**;  **position**: **relative**;  **float**: **left**;  **line-height**: 18**px**;  **text-decoration**: **none**;  **background-color**: **#fff**;  **font-size**: 14**px**;  **border**: 0;  **padding**: 9**px** 18**px**;  **color**: **#333**;  }  }   **&**.**next** {  **a** {  **background-color**: **#fafafa**;  }  }  }  }   **div** {  **color**: **#333**;  **font-size**: 14**px**;  **float**: **right**;  **width**: 241**px**;  }  }  }  }   .**hot-sale** {  **margin-bottom**: 5**px**;  **border**: 1**px solid #ddd**;   .**title** {  **font-weight**: 700;  **font-size**: 14**px**;  **line-height**: 21**px**;  **border-bottom**: 1**px solid #ddd**;  **background**: **#f1f1f1**;  **color**: **#333**;  **margin**: 0;  **padding**: 5**px** 0 5**px** 15**px**;  }   .**hot-list** {  **padding**: 15**px**;   **ul** {  **display**: **flex**;   **li** {  **width**: 25%;  **height**: 100%;   .**list-wrap** {   .**p-img**,  .**price**,  .**attr**,  .**commit** {  **padding-left**: 15**px**;  }   .**p-img** {  **img** {  **max-width**: 100%;  **vertical-align**: **middle**;  **border**: 0;  }  }   .**attr** {  **width**: 85%;  **display**: **-webkit-box**;  **-webkit-box-orient**: **vertical**;  **-webkit-line-clamp**: 2;  **overflow**: **hidden**;  **margin-bottom**: 8**px**;  **min-height**: 38**px**;  **cursor**: **pointer**;  **line-height**: 1.8;  }   .**price** {  **font-size**: 18**px**;  **color**: **#c81623**;   **strong** {  **font-weight**: 700;   **i** {  **margin-left**: -5**px**;  }  }  }   .**commit** {  **height**: 22**px**;  **font-size**: 13**px**;  **color**: **#a7a7a7**;  }  }  }  }  }  }  }  } </**style**> |

### 使用阿里的iconfont

1. 在线地址: <https://www.iconfont.cn/>
2. 注册并登陆
3. 创建一个可以包含需要的所有图标的项目
4. 搜索图标并添加到购物车
5. 将购物车中的图标添加到指定项目
6. 修改图标的名称
7. 选择Font class的使用方式, 并点击生成在线样式url
8. 在index页面中引入此图标的在线样式链接:

<link rel="stylesheet" href="http://at.alicdn.com/t/font\_1766283\_dobjk7xxze7.css">

1. 在组件中使用

<i class=”iconfont icondown”>

可以通过color改变颜色, 通过font-size改变大小

## Detail路由

### 重难点说明

1. 图片放大镜效果
2. 小图轮播

### 接口请求函数

|  |
| --- |
| *// 获取商品详情信息* **export const** *reqDetailInfo* = (skuId) => ajax.get(**`/item/**${skuId}**`**) |

### Vuex管理的详情模块

|  |
| --- |
| **import** { reqDetailInfo } **from '@/api'  const** state = {  **detailInfo**:{} *// 商品详情信息* }  **const** mutations = {  */\*   接收商品详情信息  \*/* RECEIVE\_DETAIL\_INFO (state,detailInfo){  state.**detailInfo**=detailInfo  } }  **const** actions = {  */\*   获取指定skuid的商品信息的异步action  \*/* **async** getDetailInfo({commit},skuId){  **const** result=**await** reqDetailInfo(skuId)  **if**(result.**code**===200){  **const** detailInfo = result.**data** commit(**'RECEIVE\_DETAIL\_INFO'**, detailInfo)  }  }, }  **const** getters = {  */\*   返回三级分类名称数据的对象  \*/* categoryView (state) {  **const** categoryView = state.**detailInfo**.categoryView  **return** categoryView ? categoryView : {}  },   */\*   返回商品sku相关信息对象  \*/* skuInfo(state){  **const** skuInfo = state.**detailInfo**.**skuInfo  return** skuInfo ? skuInfo : {}  },   */\*   返回商品的轮播的图片数组  \*/* skuImageList(state){  **const** skuInfo = state.**detailInfo**.**skuInfo  return** skuInfo ? skuInfo.**skuImageList** : []  },    */\*   返回商品SPU销售属性列表  \*/* spuSaleAttrList(state){  **const** spuSaleAttrList = state.**detailInfo**.spuSaleAttrList  **return** spuSaleAttrList ? spuSaleAttrList : []  } }  **export default** {  state,  actions,  mutations,  getters } |

### 商品小图片列表组件

pages/Detail/ImageList/ImageList.vue

|  |
| --- |
| <**template**>  <**div class="swiper-container" ref="swiper"**>  <**div class="swiper-wrapper"**>  <**div class="swiper-slide" v-for="**(skuImg, ***index***) in **skuImageList" :key="**skuImg.**id"**>  <**img :src="**skuImg.**imgUrl" :class="**{**active**: **currentIndex**===***index***}**"   @click="**changeCurrentIndex(***index***)**"**>  </**div**>  </**div**>  <**div class="swiper-button-next"**></**div**>  <**div class="swiper-button-prev"**></**div**>  </**div**> </**template**>  <**script**>  **import** { mapGetters } **from 'vuex'  import** Swiper **from 'swiper'  export default** {  **name**: **"ImageList"**,   data () {  **return** {  **currentIndex**: 0, *// 当前图片的下标* }  },   **computed**: {  ...mapGetters([**'skuImageList'**])  },   **watch**: {  **skuImageList**: {  handler (value) {  *// 如果图片数组长度为0, 直接结束* **if** (value.**length**==0) **return** *// 延迟到界面更新后才创建swiper对象* **this**.$nextTick(() => {  **new** Swiper(**this**.$refs.**swiper**, {  **slidesPerView**: 5, *// 一次显示5页* **slidesPerGroup**: 5, *// 以2页为单位翻页* **navigation**: { *//指定翻页按钮* **nextEl**: **'.swiper-button-next'**,  **prevEl**: **'.swiper-button-prev'**,  },  })  })   },  **immediate**: **true**, *// 初始化立即调用* }  },   **methods**: {  */\*   修改当前图片下标  \*/* changeCurrentIndex (index) {  *// 改变当前下标* **this**.**currentIndex** = index  *// 分发自定义事件, 通知父组件* **this**.$emit(**'changeCurrentIndex'**, index)  }  }  } </**script**> <**style lang="less" scoped**>  .**swiper-container** {  **height**: 56**px**;  **width**: 412**px**;  **box-sizing**: **border-box**;  **padding**: 0 12**px**;   .**swiper-slide** {  **width**: 56**px**;  **height**: 56**px**;   **img** {  **width**: 100%;  **height**: 100%;  **border**: 1**px solid #ccc**;  **padding**: 2**px**;  **width**: 50**px**;  **height**: 50**px**;  **display**: **block**;   **&**.**active** {  **border**: 2**px solid #f60**;  **padding**: 1**px**;  }   **&**:**hover** {  **border**: 2**px solid #f60**;  **padding**: 1**px**;  }  }  }   .**swiper-button-next** {  **left**: **auto**;  **right**: 0;  }   .**swiper-button-prev** {  **left**: 0;  **right**: **auto**;  }   .**swiper-button-next**,  .**swiper-button-prev** {  **box-sizing**: **border-box**;  **width**: 12**px**;  **height**: 56**px**;  **background**: **rgb**(235, 235, 235);  **border**: 1**px solid rgb**(204, 204, 204);  **top**: 0;  **margin-top**: 0;  **&**::**after** {  **font-size**: 12**px**;  }  }  } </**style**> |

### 图片放大镜组件

pages/Detail/Zoom/Zoom.vue

实现一:

|  |
| --- |
| <**template**>  <**div class="spec-preview"**>    <**img :src="imgUrl"** />    <**div class="event" @mousemove="**handleMove**" ref="event"**></**div**>    <**div class="big"**>  <**img :src="bigImgUrl" ref="bigImg"**/>  </**div**>  <**div class="mask" ref="mask"**></**div**>  </**div**> </**template**>  <**script**>  **import** throttle **from 'lodash/throttle'  export default** {  **name**: **"Zoom"**,  **props**: {  **imgUrl**: ***String***,  **bigImgUrl**: ***String*** },   **methods**: {  handleMove: throttle(**function** (event) { *// 高频调用  // 得到事件坐标* **const** {offsetX, offsetY} = event  ***console***.log(offsetX, offsetY)  *// 得到mask的宽度* **const** maskWidth = **this**.**maskWidth** *// 计算当前mask要指定left和top* **let** left = 0  **let** top = 0  left = offsetX - maskWidth/2  top = offsetY - maskWidth/2   *// left必须在[0, maskWidth]* **if** (left<0) {  left = 0  } **else if** (left>maskWidth) {  left = maskWidth  }  *// top必须在[0, maskWidth]* **if** (top<0) {  top = 0  } **else if** (top>maskWidth) {  top = maskWidth  }   *// 指定mask div的left和top样式* **const** maskDiv = **this**.$refs.mask  maskDiv.style.left = left + **'px'** maskDiv.style.top = top + **'px'** *// 指定大图 img的left和top样式* **const** bigImg = **this**.$refs.bigImg  bigImg.**style**.**left** = -2\*left + **'px'** bigImg.**style**.**top** = -2\*top + **'px'** }, 50),  },  mounted () {  *// 得到遮罩的宽度并保存* **this**.**maskWidth** = **this**.$refs.**event**.clientWidth/2 *//maskWidth是一个不变的值, 没有必要定义data中  // console.log(this.maskWidth)* }  } </**script**>  <**style lang="less"**>  .**spec-preview** {  **position**: **relative**;  **width**: 400**px**;  **height**: 400**px**;  **border**: 1**px solid #ccc**;   **img** {  **width**: 100%;  **height**: 100%;  }   .**event** {  **width**: 100%;  **height**: 100%;  **position**: **absolute**;  **top**: 0;  **left**: 0;  **z-index**: 998;  }   .**mask** {  **width**: 50%;  **height**: 50%;  **background-color**: **rgba**(0, 255, 0, 0.3);  **position**: **absolute**;  **left**: 0;  **top**: 0;  **display**: **none**;  }   .**big** {  **width**: 100%;  **height**: 100%;  **position**: **absolute**;  **top**: -1**px**;  **left**: 100%;  **border**: 1**px solid #aaa**;  **overflow**: **hidden**;  **z-index**: 998;  **display**: **none**;  **background**: **white**;   **img** {  **width**: 200%;  **max-width**: 200%;  **height**: 200%;  **position**: **absolute**;  **left**: 0;  **top**: 0;  }  }   .**event**:**hover**~.**mask**,  .**event**:**hover**~.**big** {  **display**: **block**;  }  } </**style**> |

实现二:

|  |
| --- |
| <**template**>  <**div class="spec-preview"**>  <**img :src="imgUrl"** />  <**div class="event" ref="event" @mousemove="move"**></**div**>  <**div class="big"**>  <**img :src="bigImgUrl" :style="**{**left**: **bigImgLeft**+**'px'**, **top**: **bigImgTop**+**'px'**}**"**/>  </**div**>  <**div class="mask" :style="**{**left**: **maskLeft**+**'px'**, **top**: **maskTop**+**'px'**}**"**></**div**>  </**div**> </**template**>  <**script**>  **import** throttle **from 'lodash/throttle'  export default** {  **name**: **"Zoom"**,   **props**: {  **bigImgUrl**: ***String***,  **imgUrl**: ***String*** },    data () {  **return** {  **maskLeft**: 0, *// 遮罩div的left* **maskTop**: 0, *// 遮罩div的top* **bigImgLeft**: 0, *// 大图img的left* **bigImgTop**: 0 *// 大图img的top* }  },   **methods**: {  */\*   处理鼠标移动  \*/* **move**: throttle(**function** (event) {  ***console***.log(**'-----'**)  *// 得到事件坐标(相对于中图左上角)* **const** {offsetX, offsetY} = event  *// 响应move事件div的宽度* **const** {eventWidth} = **this** *// 遮罩div的宽度* **const** maskWidth = eventWidth/2  *// 遮罩div的新的left和top值变量* **let** left = 0  **let** top = 0  *// left和top的最大值* **const** maxLeft = maskWidth  **const** maxTop = maxLeft   *// 计算left* left = offsetX - maskWidth/2  **if** (left<0) {  left = 0  } **else if** (left>maxLeft) {  left = maxLeft  }  *// 计算top* top = offsetY - maskWidth/2  **if** (top<0) {  top = 0  } **else if** (top>maxTop) {  top = maxTop  }   *// 更新遮罩div的left和top数据* **this**.**maskLeft** = left  **this**.**maskTop** = top   *// 指定大图片的left和top样式* **this**.**bigImgLeft** = -left\*2  **this**.**bigImgTop** = -top\*2  }, 20),  },   mounted() {  *// 得到响应move事件div的宽度, 并保存* **this**.**eventWidth** = **this**.**$refs**.**event**.offsetWidth *// eventWidth不需要是响应式的, 不用在data中定义* }  } </**script**>  <**style lang="less"**>  .**spec-preview** {  **position**: **relative**;   **img** {  **width**: 100%;  **height**: 100%;  }   .**event** {  **width**: 100%;  **height**: 100%;  **position**: **absolute**;  **top**: 0;  **left**: 0;  **z-index**: 999;  }   .**mask** {  **width**: 50%;  **height**: 50%;  **background-color**: **rgba**(0, 255, 0, 0.3);  **position**: **absolute**;  **left**: 0;  **top**: 0;  **display**: **none**;  }   .**big** {  **width**: 100%;  **height**: 100%;  **position**: **absolute**;  **top**: -1**px**;  **left**: 100%;  **border**: 1**px solid #ccc**;  **overflow**: **hidden**;  **z-index**: 998;  **display**: **none**;   **img** {  **width**: 200%;  **max-width**: 200%;  **height**: 200%;  **position**: **absolute**;  **left**: 0;  **top**: 0;  }  }   .**event**:**hover**~.**mask**,  .**event**:**hover**~.**big** {  **display**: **block**;  }  } </**style**> |

### Detail路由组件

pages/Detail/index.vue

|  |
| --- |
| <**template**>  <**div class="detail"**>  *<!-- 商品分类导航 -->* <**TypeNav** />   *<!-- 主要内容区域 -->* <**section class="con"**>  *<!-- 导航路径区域 -->* <**div class="conPoin"**>  <**span**>{{categoryView.category1Name}}</**span**>  <**span**>{{categoryView.category2Name}}</**span**>  <**span**>{{categoryView.category3Name}}</**span**>  </**div**>  *<!-- 主要内容区域 -->* <**div class="mainCon"**>  *<!-- 左侧放大镜区域 -->* <**div class="previewWrap"**>  *<!--放大镜效果-->* <**Zoom   v-if="skuImageList**.**length**>0**"  :imgUrl="skuImageList**[**currentImgIndex**].**imgUrl"   :bigImgUrl="skuImageList**[**currentImgIndex**].**imgUrl"**/>  *<!-- 小图列表 -->* <**ImageList @changeCurrentIndex="**changeCurrentIndex**"**/>  </**div**>  *<!-- 右侧选择区域布局 -->* <**div class="InfoWrap"**>  <**div class="goodsDetail"**>  <**h3 class="InfoName"**>{{skuInfo.skuName}}</**h3**>  <**p class="news"**>{{skuInfo.skuDesc}}</**p**>  <**div class="priceArea"**>  <**div class="priceArea1"**>  <**div class="title"**>价**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**格</**div**>  <**div class="price"**>  <**i**>¥</**i**>  <**em**>{{skuInfo.price}}</**em**>  <**span**>降价通知</**span**>  </**div**>  <**div class="remark"**>  <**i**>累计评价</**i**>  <**em**>65545</**em**>  </**div**>  </**div**>  <**div class="priceArea2"**>  <**div class="title"**>  <**i**>促**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**销</**i**>  </**div**>  <**div class="fixWidth"**>  <**i class="red-bg"**>加价购</**i**>  <**em class="t-gray"**>满999.00另加20.00元，或满1999.00另加30.00元，或满2999.00另加40.00元，即可在购物车换购热销商品</**em**>  </**div**>  </**div**>  </**div**>  <**div class="support"**>  <**div class="supportArea"**>  <**div class="title"**>支**&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;**持</**div**>  <**div class="fixWidth"**>以旧换新，闲置手机回收 4G套餐超值抢 礼品购</**div**>  </**div**>  <**div class="supportArea"**>  <**div class="title"**>配 送 至</**div**>  <**div class="fixWidth"**>广东省 深圳市 宝安区</**div**>  </**div**>  </**div**>  </**div**>   <**div class="choose"**>  <**div class="chooseArea"**>  <**div class="choosed"**></**div**>  <**dl v-for="**(**attr**, ***index***) in spuSaleAttrList**" :key="attr**.**id"**>  <**dt class="title"**>{{**attr**.saleAttrName}}</**dt**>  <**dd   v-for="**(***value***, ***index***) in **attr**.spuSaleAttrValueList**"   :key="*value***.**id" :class="**{**active**: ***value***.**isChecked**===**'1'**}**"   @click="**selectValue(***value***, **attr**.spuSaleAttrValueList)**"** >  {{***value***.saleAttrValueName}}  </**dd**>  </**dl**>  </**div**>  <**div class="cartWrap"**>  <**div class="controls"**>  <**input autocomplete="off" class="itxt" v-model="skuNum"**>  <**a href="javascript:" class="plus" @click="skuNum**++**"**>+</**a**>  <**a href="javascript:" class="mins" @click="skuNum**>1 ? **skuNum**-- : **''"**>-</**a**>  </**div**>  <**div class="add"**>  <**a href="javascript:" @click="**addToCart**"**>加入购物车</**a**>  </**div**>  </**div**>  </**div**>  </**div**>  </**div**>  </**section**>   *<!-- 内容详情页 -->* <**section class="product-detail"**>  <**aside class="aside"**>  <**div class="tabWraped"**>  <**h4 class="active"**>相关分类</**h4**>  <**h4**>推荐品牌</**h4**>  </**div**>  <**div class="tabContent"**>  <**div class="tab-pane active"**>  <**ul class="partList"**>  <**li**>手机</**li**>  <**li**>手机壳</**li**>  <**li**>内存卡</**li**>  <**li**>Iphone配件</**li**>  <**li**>贴膜</**li**>  <**li**>手机耳机</**li**>  <**li**>移动电源</**li**>  <**li**>平板电脑</**li**>  </**ul**>  <**ul class="goodsList"**>  <**li**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**img src="./images/part01.png"** />  </**div**>  <**div class="attr"**>Apple苹果iPhone 6s (A1699) </**div**>  <**div class="price"**>  <**em**>¥</**em**>  <**i**>6088.00</**i**>  </**div**>  <**div class="operate"**>  <**a href="javascript:void(0);"**>加入购物车</**a**>  </**div**>  </**div**>  </**li**>  <**li**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**img src="./images/part02.png"** />  </**div**>  <**div class="attr"**>  <**em**>Apple苹果iPhone 6s (A1699)</**em**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>6088.00</**i**>  </**strong**>  </**div**>  <**div class="operate"**>  <**a href="javascript:void(0);"**>加入购物车</**a**>  </**div**>  </**div**>  </**li**>  <**li**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**img src="./images/part03.png"** />  </**div**>  <**div class="attr"**>  <**em**>Apple苹果iPhone 6s (A1699)</**em**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>6088.00</**i**>  </**strong**>  </**div**>  <**div class="operate"**>  <**a href="javascript:void(0);"**>加入购物车</**a**>  </**div**>  </**div**>  </**li**>  <**li**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**img src="./images/part02.png"** />  </**div**>  <**div class="attr"**>  <**em**>Apple苹果iPhone 6s (A1699)</**em**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>6088.00</**i**>  </**strong**>  </**div**>  <**div class="operate"**>  <**a href="javascript:void(0);"**>加入购物车</**a**>  </**div**>  </**div**>  </**li**>  <**li**>  <**div class="list-wrap"**>  <**div class="p-img"**>  <**img src="./images/part03.png"** />  </**div**>  <**div class="attr"**>  <**em**>Apple苹果iPhone 6s (A1699)</**em**>  </**div**>  <**div class="price"**>  <**strong**>  <**em**>¥</**em**>  <**i**>6088.00</**i**>  </**strong**>  </**div**>  <**div class="operate"**>  <**a href="javascript:void(0);"**>加入购物车</**a**>  </**div**>  </**div**>  </**li**>  </**ul**>  </**div**>  <**div class="tab-pane"**>  <**p**>推荐品牌</**p**>  </**div**>  </**div**>  </**aside**>  <**div class="detail"**>  <**div class="fitting"**>  <**h4 class="kt"**>选择搭配</**h4**>  <**div class="good-suits"**>  <**div class="master"**>  <**img src="./images/l-m01.png"** />  <**p**>￥5299</**p**>  <**i**>+</**i**>  </**div**>  <**ul class="suits"**>  <**li class="suitsItem"**>  <**img src="./images/dp01.png"** />  <**p**>Feless费勒斯VR</**p**>  <**label**>  <**input type="checkbox" value="39"**>  <**span**>39</**span**>  </**label**>  </**li**>  <**li class="suitsItem"**>  <**img src="./images/dp02.png"** />  <**p**>Feless费勒斯VR</**p**>  <**label**>  <**input type="checkbox" value="50"**>  <**span**>50</**span**>  </**label**>  </**li**>  <**li class="suitsItem"**>  <**img src="./images/dp03.png"** />  <**p**>Feless费勒斯VR</**p**>  <**label**>  <**input type="checkbox" value="59"**>  <**span**>59</**span**>  </**label**>  </**li**>  <**li class="suitsItem"**>  <**img src="./images/dp04.png"** />  <**p**>Feless费勒斯VR</**p**>  <**label**>  <**input type="checkbox" value="99"**>  <**span**>99</**span**>  </**label**>  </**li**>  </**ul**>  <**div class="result"**>  <**div class="num"**>已选购0件商品</**div**>  <**div class="price-tit"**>  套餐价  </**div**>  <**div class="price"**>￥5299</**div**>  <**button class="addshopcar"**>加入购物车</**button**>  </**div**>  </**div**>  </**div**>  <**div class="intro"**>  <**ul class="tab-wraped"**>  <**li class="active"**>  <**a href="###"**>  商品介绍  </**a**>  </**li**>  <**li**>  <**a href="###"**>  规格与包装  </**a**>  </**li**>  <**li**>  <**a href="###"**>  售后保障  </**a**>  </**li**>  <**li**>  <**a href="###"**>  商品评价  </**a**>  </**li**>  <**li**>  <**a href="###"**>  手机社区  </**a**>  </**li**>  </**ul**>  <**div class="tab-content"**>  <**div id="one" class="tab-pane active"**>  <**ul class="goods-intro"**>  <**li**>分辨率：1920\*1080(FHD)</**li**>  <**li**>后置摄像头：1200万像素</**li**>  <**li**>前置摄像头：500万像素</**li**>  <**li**>核 数：其他</**li**>  <**li**>频 率：以官网信息为准</**li**>  <**li**>品牌： Apple</**li**>  <**li**>商品名称：APPLEiPhone 6s Plus</**li**>  <**li**>商品编号：1861098</**li**>  <**li**>商品毛重：0.51kg</**li**>  <**li**>商品产地：中国大陆</**li**>  <**li**>热点：指纹识别，Apple Pay，金属机身，拍照神器</**li**>  <**li**>系统：苹果（IOS）</**li**>  <**li**>像素：1000-1600万</**li**>  <**li**>机身内存：64GB</**li**>  </**ul**>  <**div class="intro-detail"**>  <**img src="./images/intro01.png"** />  <**img src="./images/intro02.png"** />  <**img src="./images/intro03.png"** />  </**div**>  </**div**>  <**div id="two" class="tab-pane"**>  <**p**>规格与包装</**p**>  </**div**>  <**div id="three" class="tab-pane"**>  <**p**>售后保障</**p**>  </**div**>  <**div id="four" class="tab-pane"**>  <**p**>商品评价</**p**>  </**div**>  <**div id="five" class="tab-pane"**>  <**p**>手机社区</**p**>  </**div**>  </**div**>  </**div**>  </**div**>  </**section**>  </**div**> </**template**>  <**script**>  **import** {mapGetters} **from 'vuex'  import** ImageList **from './ImageList/ImageList'  import** Zoom **from './Zoom/Zoom'   export default** {  **name**: **'Detail'**,   data () {  **return** {  **currentImgIndex**: 0, *// 当前图片下标* **skuNum**: 1, *// 商品的数量* }  },   mounted () {  *// 取出skuId的params参数* **const** skuId = **this**.$route.params.skuId  *// 分发给获取商品详情的异步action* **this**.$store.**dispatch**(**'getDetailInfo'**, skuId)  },   **computed**: {  ...mapGetters([**'categoryView'**, **'skuInfo'**, **'spuSaleAttrList'**, **'skuImageList'**])  },   **methods**: {  */\*   选择某个属性值  \*/* selectValue (value, valueList) {  *// 如果当前项没有选中才处理* **if** (value.**isChecked**!==**'1'**) {  *// 将所有的项都先指定为不选择* valueList.forEach(v => v.**isChecked** = **'0'**)  *// 选中当前的* value.**isChecked** = **'1'** }  },   */\*   changeCurrentIndex事件的回调函数  \*/* changeCurrentIndex (index) {  *// 更新当前的currentImgIndex* **this**.**currentImgIndex** = index  },   */\*   将当前商品添加到购物车, 成功后跳转到成功界面  \*/* **async** addToCart () {  **const** query = {**skuId**: **this**.**skuInfo**.**id**, **skuNum**: **this**.**skuNum**}  *// 分发添加购物车的action  // this.$store.dispatch('addToCart', {...query, callback: this.callback})* **const** errorMsg = **await this**.$store.**dispatch**(**'addToCart2'**, query)  **this**.callback(errorMsg)  },   callback (errorMsg) {   **const** query = {**skuId**: **this**.**skuInfo**.**id**, **skuNum**: **this**.**skuNum**}  *// 如果成功了* **if** (!errorMsg) {    *// 在跳转前将skuInfo保存到sessionStorage (key=value, value只能是字符串)* **window**.**sessionStorage**.setItem(**'SKU\_INFO\_KEY'**, ***JSON***.stringify(**this**.**skuInfo**))   **this**.$router.push({**path**: **'/addcartsuccess'**, query})  } **else** {  *alert*(errorMsg)  }  }  },    **components**: {  ImageList,  Zoom  }  } </**script**>  <**style lang="less" scoped**>  .**detail** {  .**con** {  **width**: 1200**px**;  **margin**: 15**px auto** 0;   .**conPoin** {  **padding**: 9**px** 15**px** 9**px** 0;   **&**>**span**+**span**:**before** {  **content**: **"/\00a0"**;  **padding**: 0 5**px**;  **color**: **#ccc**;  }  }   .**mainCon** {  **overflow**: **hidden**;  **margin**: 5**px** 0 15**px**;   .**previewWrap** {  **float**: **left**;  **width**: 400**px**;  **position**: **relative**;  }   .**InfoWrap** {  **width**: 700**px**;  **float**: **right**;   .**InfoName** {  **font-size**: 14**px**;  **line-height**: 21**px**;  **margin-top**: 15**px**;  }   .**news** {  **color**: **#e12228**;  **margin-top**: 15**px**;  }   .**priceArea** {  **background**: **#fee9eb**;  **padding**: 7**px**;  **margin**: 13**px** 0;   .**priceArea1** {  **overflow**: **hidden**;  **line-height**: 28**px**;  **margin-top**: 10**px**;   .**title** {  **float**: **left**;  **margin-right**: 15**px**;  }   .**price** {  **float**: **left**;  **color**: **#c81623**;   **i** {  **font-size**: 16**px**;  }   **em** {  **font-size**: 24**px**;  **font-weight**: 700;  }   **span** {  **font-size**: 12**px**;  }  }   .**remark** {  **float**: **right**;  }  }   .**priceArea2** {  **overflow**: **hidden**;  **line-height**: 28**px**;  **margin-top**: 10**px**;   .**title** {  **margin-right**: 15**px**;  **float**: **left**;  }   .**fixWidth** {  **width**: 520**px**;  **float**: **left**;   .**red-bg** {  **background**: **#c81623**;  **color**: **#fff**;  **padding**: 3**px**;  }   .**t-gray** {  **color**: **#999**;  }  }  }    }   .**support** {  **border-bottom**: 1**px solid #ededed**;  **padding-bottom**: 5**px**;   .**supportArea** {  **overflow**: **hidden**;  **line-height**: 28**px**;  **margin-top**: 10**px**;   .**title** {  **margin-right**: 15**px**;  **float**: **left**;  }   .**fixWidth** {  **width**: 520**px**;  **float**: **left**;  **color**: **#999**;  }  }  }   .**choose** {  .**chooseArea** {  **overflow**: **hidden**;  **line-height**: 28**px**;  **margin-top**: 10**px**;   **dl** {  **overflow**: **hidden**;  **margin**: 13**px** 0;   **dt** {  **margin-right**: 15**px**;  **float**: **left**;  }   **dd** {  **float**: **left**;  **margin-right**: 5**px**;  **color**: **#666**;  **line-height**: 24**px**;  **padding**: 2**px** 14**px**;  **border-top**: 1**px solid #eee**;  **border-right**: 1**px solid #bbb**;  **border-bottom**: 1**px solid #bbb**;  **border-left**: 1**px solid #eee**;   **&**.**active** {  **color**: **green**;  **border**: 1**px solid green**;  }  }  }  }   .**cartWrap** {  .**controls** {  **width**: 48**px**;  **position**: **relative**;  **float**: **left**;  **margin-right**: 15**px**;   .**itxt** {  **width**: 38**px**;  **height**: 37**px**;  **border**: 1**px solid #ddd**;  **color**: **#555**;  **float**: **left**;  **border-right**: 0;  **text-align**: **center**;  }   .**plus**,  .**mins** {  **width**: 15**px**;  **text-align**: **center**;  **height**: 17**px**;  **line-height**: 17**px**;  **background**: **#f1f1f1**;  **color**: **#666**;  **position**: **absolute**;  **right**: -8**px**;  **border**: 1**px solid #ccc**;  }   .**mins** {  **right**: -8**px**;  **top**: 19**px**;  **border-top**: 0;  }   .**plus** {  **right**: -8**px**;  }  }   .**add** {  **float**: **left**;   **a** {  **background-color**: **#e1251b**;  **padding**: 0 25**px**;  **font-size**: 16**px**;  **color**: **#fff**;  **height**: 36**px**;  **line-height**: 36**px**;  **display**: **block**;  }  }  }  }  }  }  }   .**product-detail** {  **width**: 1200**px**;  **margin**: 30**px auto** 0;  **overflow**: **hidden**;   .**aside** {  **width**: 210**px**;  **float**: **left**;  **border**: 1**px solid #ccc**;   .**tabWraped** {  **height**: 40**px**;   **h4** {  **border-top**: 3**px solid #fff**;  **float**: **left**;  **line-height**: 37**px**;  **width**: 105**px**;  **text-align**: **center**;  **border-bottom**: 1**px solid #ccc**;   **&**.**active** {  **border-top**: 3**px solid #e1251b**;  **border-bottom**: 0;  **font-weight**: **normal**;  }  }  }   .**tabContent** {  **padding**: 10**px**;   .**tab-pane** {  **display**: **none**;   **&**.**active** {  **display**: **block**;  }   **&**:**nth-child**(1) {  .**partList** {  **overflow**: **hidden**;   **li** {  **width**: 50%;  **float**: **left**;  **border-bottom**: 1**px dashed #ededed**;  **line-height**: 28**px**;  }  }   .**goodsList** {  **&**>**li** {  **margin**: 5**px** 0 15**px**;  **border-bottom**: 1**px solid #ededed**;  **padding-bottom**: 5**px**;   .**list-wrap** {  .**p-img** {  **text-align**: **center**;   **img** {  **width**: 152**px**;  }  }   .**price** {  **font-size**: 16**px**;  **color**: **#c81623**;  }   .**operate** {  **text-align**: **center**;  **margin**: 5**px** 0;   **a** {  **background-color**: **transparent**;  **border**: 1**px solid #8c8c8c**;  **color**: **#8c8c8c**;  **display**: **inline-block**;  **padding**: 2**px** 14**px**;  **line-height**: 18**px**;  }  }  }  }  }  }  }  }  }   .**detail** {  **width**: 980**px**;  **float**: **right**;   .**fitting** {  **border**: 1**px solid #ddd**;  **margin-bottom**: 15**px**;   .**kt** {  **border-bottom**: 1**px solid #ddd**;  **background**: **#f1f1f1**;  **color**: **#333**;  **padding**: 5**px** 0 5**px** 15**px**;  }   .**good-suits** {  **height**: 170**px**;  **padding-top**: 10**px**;   .**master** {  **width**: 127**px**;  **height**: 165**px**;  **text-align**: **center**;  **position**: **relative**;  **float**: **left**;   **img** {  **width**: 87**px**;  }   **p** {  **color**: **#c81623**;  **font-size**: 16**px**;  **font-weight**: 700;  }   **i** {  **position**: **absolute**;  **top**: 48**px**;  **right**: -25**px**;  **font-size**: 16**px**;  }  }   .**suits** {  **width**: 668**px**;  **height**: 165**px**;  **float**: **left**;   .**suitsItem** {  **float**: **left**;  **width**: 127**px**;  **padding**: 0 20**px**;  **text-align**: **center**;   **img** {  **width**: 120**px**;  **height**: 130**px**;  }   **p** {  **font-size**: 12**px**;  }   **label** {  **display**: **block**;  **position**: **relative**;   **input** {  **vertical-align**: **middle**;  }   **span** {  **vertical-align**: **middle**;  }  }  }  }   .**result** {  **border-left**: 1**px solid #ddd**;  **width**: 153**px**;  **height**: 165**px**;  **padding-left**: 20**px**;  **float**: **left**;   .**num** {  **font-size**: 14**px**;  **margin-bottom**: 10**px**;  **margin-top**: 10**px**;  }   .**price-tit** {  **font-weight**: **bold**;  **margin-bottom**: 10**px**;  }   .**price** {  **color**: **#B1191A**;  **font-size**: 16**px**;  **margin-bottom**: 10**px**;  }   .**addshopcar** {  **background-color**: **#e1251b**;  **border**: 1**px solid #e1251b**;  **padding**: 10**px** 25**px**;  **font-size**: 16**px**;  **color**: **#fff**;  **display**: **inline-block**;  **box-sizing**: **border-box**;  }  }  }  }   .**intro** {  .**tab-wraped** {  **background**: **#ededed**;  *// border: 1px solid #ddd;* **overflow**: **hidden**;   **li** {  **float**: **left**;   **&**+**li**>**a** {  **border-left**: 1**px solid #ddd**;  }   **&**.**active** {  **a** {  *// border: 0;* **background**: **#e1251b**;  **color**: **#fff**;  }  }   **a** {  **display**: **block**;  **height**: 40**px**;  **line-height**: 40**px**;  **padding**: 0 11**px**;  **text-align**: **center**;  **color**: **#666**;  **background**: **#fcfcfc**;  **border-top**: 1**px solid #ddd**;  **border-bottom**: 1**px solid #ddd**;  }  }  }   .**tab-content** {  .**tab-pane** {  **display**: **none**;   **&**.**active** {  **display**: **block**;  }   **&**:**nth-child**(1) {  .**goods-intro** {  **padding-left**: 10**px**;   **li** {  **margin**: 10**px** 0;  }  }   .**intro-detail** {  **img** {  **width**: 100%;  }  }  }  }   }  }  }  }  } </**style**> |

### 注册Detail路由

router/routes.js

|  |
| --- |
| {  **path**: **'/detail/:skuId'**,  **component**: Detail }, |

## AddCartSuccess路由

### 重难点说明

1. 区别使用sessionStorage与localStorage

### 跳转到此路由

|  |
| --- |
| */\*  添加到购物车 \*/* **async** addToCart() {  **const** { skuId } = **this**.$route.params  **const** {skuNum} = **this  window**.**sessionStorage**.setItem(**'SKU\_INFO'**, ***JSON***.stringify(**this**.skuInfo))  **const** message = **await this**.$store.**dispatch**(**'addToCart'**, {skuId,skuNum})  **if** (message) { *// 有错误message, 提示添加购物失败  alert*(message)  } **else** { *// 添加购物车成功, 跳转到成功界面* **this**.$router.push({  **path**: **'/addcartsuccess'**,  **query**: {skuNum }  })  } } |

### AddCartSuccess路由组件

|  |
| --- |
| <**template**>  <**div class="cart-complete-wrap"**>  <**div class="cart-complete"**>  <**h3**><**i class="sui-icon icon-pc-right"**></**i**>商品已成功加入购物车！</**h3**>  <**div class="goods"**>  <**div class="left-good"**>  <**div class="left-pic"**>  <**img :src="skuInfo**.skuDefaultImg**"**>  </**div**>  <**div class="right-info"**>  <**p class="title"**>{{ **skuInfo**.skuName }}</**p**>  <**p class="attr"**>颜色：WFZ5099IH/5L钛金釜内胆 数量：{{ **skuNum** }}</**p**>  </**div**>  </**div**>  <**div class="right-gocart"**>  <**a href="javascript:" class="sui-btn btn-xlarge"   @click="**toDetail**"**>查看商品详情2</**a**>  <**router-link to="/shopcart"**>去购物车结算 > </**router-link**>  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **'AddCartSuccess'**,  **props**: [**'skuId'**, **'skuNum'**],  data() {  **return** {  **skuInfo**: {}  }  },    mounted() {  **this**.**skuInfo** = ***JSON***.parse(**window**.**sessionStorage**.getItem(**'SKU\_INFO'**))  },   **methods**: {  *// 跳转到详情页面* toDetail () {  **this**.$router.push({ **name**: **'detail'**, **params**: {**skuId**: **this**.**skuId**} })  }  }  } </**script**> |

## ShopCart路由

### 重难点说明

1. 用户临时ID的处理
2. 购物车数据的管理(复杂)
3. 不使用v-model监控用户输入
4. async / await / Promise.all() 的使用

### 接口请求函数

|  |
| --- |
| *// 添加到购物车(修改购物项数量) // skuNum指定为改变的数量, 如果是减少就是负数* **export const** *reqAddToCart* = (skuId, skuNum) => ajax.post(**`/cart/addToCart/**${skuId}**/**${skuNum}**`**) *// 获取购物车数据列表* **export const** *reqCartList* = () => ajax.get(**'/cart/cartList'**) *// 指定购物项的选中状态 /api/cart/checkCart/{skuID}/{isChecked}* **export const** *reqCheckCartItem* = (skuId, isChecked) => ajax.get(**`/cart/checkCart/**${skuId}**/**${isChecked}**`**) *// 删除购物车商品 /api/cart/deleteCart/{skuId}* **export const** *reqDeleteCartItem* = (skuId) => ajax.delete(**`/cart/deleteCart/**${skuId}**`**) |

### vuex管理的购物车模块

store/modules/shopCart.js

|  |
| --- |
| */\*  管理购物车模块相关数据的vuex模块 \*/* **import** { reqAddToCart, reqCartList, reqCheckCartItem, reqDeleteCartItem } **from "@/api"  const** state = {  **cartList**: [], *// 所有购物车列表数据* } **const** mutations = {  RECEIVE\_CART\_LIST (state, cartList) {  state.**cartList** = cartList  }  } **const** actions = {   */\*   设置购物项的选中状态  \*/* **async** checkCartItem ({commit}, {skuId, isChecked}) {  **const** result = **await** reqCheckCartItem(skuId, isChecked)  **if** (result.**code**!==200) {  **throw new** Error(**'勾选购物项失败'**)  }  },   */\*   删除某个购物项的异步action  \*/* **async** deleteCartItem ({commit}, skuId) {  **const** result = **await** reqDeleteCartItem(skuId)  */\* if (result.code===200) {  // 成功了, 告诉组件重新获取购物车列表数据   } else {// 失败了, 告诉组件显示失败的提示   } \*/* **return** result.**code**===200 ? **''** : result.**message** || **'删除购物项失败'** },    **async** deleteCartItem2 ({commit}, skuId) {  **const** result = **await** reqDeleteCartItem(skuId)  */\* if (result.code===200) {  // 成功了, 告诉组件重新获取购物车列表数据   } else {// 失败了, 告诉组件显示失败的提示   } \*/  // return result.code===200 ? '' : result.message || '删除购物项失败'* **if** (result.**code**!==200) { *// 如果删除失败, 抛出一个error, error包含要显示的提示文本* **throw new** Error(**'删除购物项失败'**)  *// return Promise.reject(new Error('删除购物项失败'))  // return new Error('删除购物项失败') 不可以* }  },   */\*   context对象:  {  state, // 等同于 `store.state`，若在模块中则为局部状态  rootState, // 等同于 `store.state`，只存在于模块中  commit, // 等同于 `store.commit`  dispatch, // 等同于 `store.dispatch`  getters, // 等同于 `store.getters`  rootGetters // 等同于 `store.getters`，只存在于模块中  }  \*/* **async** deleteCartItems ({dispatch, getters}) {  **const** promises = []  *// 删除selectedItems所有的购物项* getters.selectedItems.forEach(item => {  *// 删除item购物项* **const** promise = dispatch(**'deleteCartItem2'**, item.**skuId**)  *// 保存返回的promise* promises.push(promise)  })   *// 如果都成功了, 才去重新获取新的购物车数据* **return** Promise.all(promises)  },   */\*   获取购物车数据列表的异步action  \*/* **async** getCartList ({commit}) {  **const** result = **await** reqCartList()  **if** (result.**code**===200) {  **const** cartList = result.**data** commit(**'RECEIVE\_CART\_LIST'**, cartList)  }  },   */\*   添加到购物车的异步action  \*/* **async** addToCart ({commit}, {skuId, skuNum, callback}) {  **const** result = **await** reqAddToCart(skuId, skuNum)  **if** (result.**code**===200) { *// 成功* callback(**''**)  } **else** { *// 失败* callback(result.**message** || **'添加购物车失败'**)  }  },   **async** addToCart2 ({commit}, {skuId, skuNum}) {  **const** result = **await** reqAddToCart(skuId, skuNum)  **return** result.**code**===200 ? **''** : (result.**message** || **'添加购物车失败'**)  },   **async** addToCart3 ({dispatch}, {skuId, skuNum}) {  **const** result = **await** reqAddToCart(skuId, skuNum)  **if** (result.**code**===200) { *// 成功了  // 分发请求获取最新购物车列表的action* dispatch(**'getCartList'**)  } **else** { *// 失败了  // 可以直接提示* alert(result.**message** || **'添加购物车失败'**)  *// 选择让组件处理错误提示  // return (result.message || '添加购物车失败')* }  } } **const** getters = {  */\*   总数量  \*/* totalCount (state) {  */\* let total = 0  state.cartList.forEach((item, index) => {  if (item.isChecked===1) {  total += item.skuNum  }  })   return total  \*/    // 使用reduce进行累计/累加的操作* **return** state.**cartList**.reduce((pre, item) => {  */\*   if (item.isChecked===1) {  return pre + item.skuNum  } else {  return pre  } \*/* **return** item.**isChecked**===1 ? pre + item.**skuNum** : pre  } , 0)  },   */\*   总价格  \*/* totalPrice (state) {  **return** state.**cartList**.reduce((pre, item) => {  **return** item.**isChecked**===1 ? pre + item.**skuNum**\*item.cartPrice : pre  } , 0)  },    */\*   是否全选  \*/* isAllChecked (state) {  *// arr.every(): 判断所有的元素是否都满足条件* **return** state.**cartList**.**length**>0 && state.**cartList**.every((item , index) => item.**isChecked**===1)  },   */\*   所有选中购物项的数组  \*/* selectedItems (state) {  *// filter(): 返回所有满足条件的数组元素组成的新数组  // return state.cartList.filter((item, index) => item.isChecked===1)  // reduce(): 返回累计累加后的最后结果* **return** state.**cartList**.reduce((pre, item) => {  **if** (item.**isChecked**===1) {  pre.push(item)  }  **return** pre *// 就是初始指定的数组, 只是很可能前面添加了item元素* }, [])  } }  **export default** {  state,  mutations,  actions,  getters } |

### 请求携带唯一的用户临时ID

1. 下载工具包

npm install uuid store -S

1. utils/storageUtils

|  |
| --- |
| */\*  localStorage存储管理的工具函数模块 \*/* **import** store **from 'store' import** UUID **from 'uuidjs'  const** USER\_TEMP\_ID\_KEY = **'user\_temp\_id\_key'** */\*  获取用户临时ID \*/* **export function** *getUserTempId* () {  *// 从localStorage中读取* **let** userTempId = store.get(USER\_TEMP\_ID\_KEY)  *// 如果localStorage中没有, 生成一个新的, 并保存到local中* **if** (!userTempId) {  userTempId = UUID.*generate*()  store.set(USER\_TEMP\_ID\_KEY, userTempId)  }  *// 返回它* **return** userTempId } |

1. store/modules/user.js

|  |
| --- |
| **import** {getUserTempId} **from '@/utils/storageUtils'** */\*  用于操作首页模块数据的vuex模块 \*/* **const** state = {  **userInfo**: {}, *// 登陆的用户信息对象* **userTempId**: getUserTempId(), *// 登陆前的唯一标识, 发送请求时需要携带* } **const** mutations = {} **const** actions = {} **const** getters = {}  **export default** {  state,  mutations,  actions,  getters } |

1. api/ajax.js

|  |
| --- |
| *// 添加请求拦截器* ajax.interceptors.request.use((config) => {  */\* 2. 显示请求进度条 \*/  // 显示进度条* NProgress.start()   */\* 5. 每次请求总是携带用户临时ID(不管是否登陆) \*/* config.headers[**'userTempId'**] = store.**state**.**user**.**userTempId** *// 必须返回config* **return** config }) |

### ShopCart路由组件

pages/ShopCart/index.vue

|  |
| --- |
| <**template**>  <**div class="cart"**>  <**h4**>全部商品</**h4**>  <**div class="cart-main"**>  <**div class="cart-th"**>  <**div class="cart-th1"**>全部</**div**>  <**div class="cart-th2"**>商品</**div**>  <**div class="cart-th3"**>单价（元）</**div**>  <**div class="cart-th4"**>数量</**div**>  <**div class="cart-th5"**>小计（元）</**div**>  <**div class="cart-th6"**>操作</**div**>  </**div**>  <**div class="cart-body"**>    <**ul class="cart-list" v-for="**(***item***, ***index***) in **cartList" :key="*item***.**id"**>  <**li class="cart-list-con1"**>  <**input type="checkbox" name="chk\_list" @change="**checkItem(***item***)**" :checked="*item***.**isChecked"**>  </**li**>  <**li class="cart-list-con2"**>  <**img :src="*item***.**imgUrl"**>  <**div class="item-msg"**>{{***item***.skuName}}</**div**>  </**li**>  <**li class="cart-list-con3"**>  <**div class="item-txt"**>语音升级款</**div**>  </**li**>  <**li class="cart-list-con4"**>  <**span class="price"**>{{***item***.skuPrice}}</**span**>  </**li**>  <**li class="cart-list-con5"**>  <**a href="javascript:void(0)" class="mins" @click="**changeItemNum(***item***, -1)**"**>-</**a**>  <**input autocomplete="off" type="text" class="itxt"   @change="**changeItemNum(***item***, $event.**target**.**value**\*1-***item***.**skuNum**)**" :value="*item***.**skuNum"**>  <**a href="javascript:void(0)" class="plus" @click="**changeItemNum(***item***, 1)**"**>+</**a**>  </**li**>  <**li class="cart-list-con6"**>  <**span class="sum"**>{{***item***.skuPrice \* ***item***.**skuNum**}}</**span**>  </**li**>  <**li class="cart-list-con7"**>  <**a href="#none" class="sindelet" @click="**deleteItem(***item***)**"**>删除</**a**>  <**br**>  <**a href="#none"**>移到收藏</**a**>  </**li**>  </**ul**>  </**div**>  </**div**>  <**div class="cart-tool"**>  <**div class="select-all"**>  <**input class="chooseAll" type="checkbox" v-model="allChecked"**>  <**span**>全选</**span**>  </**div**>  <**div class="option"**>  <**a href="javascript:" @click="**deleteCheckedItems**"**>删除选中的商品</**a**>  <**a href="javascript:"**>移到我的关注</**a**>  <**a href="javascript:"**>清除下柜商品</**a**>  </**div**>  <**div class="money-box"**>  <**div class="chosed"**>已选择  <**span**>{{totalCount}}</**span**>件商品</**div**>  <**div class="sumprice"**>  <**em**>总价（不含运费） ：</**em**>  <**i class="summoney"**>{{totalPrice}}</**i**>  </**div**>  <**div class="sumbtn"**>  <**a class="sum-btn" href="javascript:"**>结算</**a**>  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **import** {  mapState,  mapGetters  } **from 'vuex'  export default** {  **name**: **'ShopCart'**,   **computed**: {  ...mapState({  cartList: state => state.**shopCart**.**cartList** }),   ...mapGetters([**'totalPrice'**, **'totalCount'**, **'checkedCartItems'**, **'isAllChecked'**]),   **allChecked**: {  get () {  **return this**.isAllChecked  },   */\*   当改变勾选, 请求更新所有购物项的的勾选状态  \*/* **async** set (value) {  **const** promises = **this**.**cartList**.map(item => {  **return this**.$store.**dispatch**(**'checkCartItem'**, {**skuId**: item.skuId,**isChecked**: value \* 1})  })  **await** Promise.all(promises)  **this**.getCartList()  }  }  },   mounted() {  **this**.getCartList()  },   **methods**: {  */\*   异步获取购物车数据列表  \*/* getCartList () {  **this**.$store.**dispatch**(**'getCartList'**)  },   */\*   异步改变购物项的数量  \*/* **async** changeItemNum (item, numChange) {  **const** {skuNum, skuId} = item  **if** (skuNum + numChange > 0) {   **await this**.$store.**dispatch**(**'addToCart'**, {skuId,**skuNum**: numChange})  **this**.getCartList()  }  },   */\*   异步改变购物项的勾选状态  \*/* **async** checkItem (item) {  **let** {skuId, isChecked} = item  isChecked = isChecked===1 ? 0 : 1  **try** {  **await this**.$store.**dispatch**(**'checkCartItem'**, {skuId,isChecked})  **this**.getCartList()  } **catch** (error) {  *alert*(error.**message** || **'勾选购物项失败'**)  }    },   */\*   异步删除指定购物项  \*/* **async** deleteItem (item) {  **if** (**window**.confirm(**'确定要删除吗?'**)) {  **try** {  **await this**.$store.**dispatch**(**'deleteCartItem'**, item.skuId)  **this**.getCartList()  } **catch** (error) {  *alert*(error.**message** || **'删除购物项失败'**)  }  }  },   */\*   删除所有选中的购物项  \*/* **async** deleteCheckedItems () {  **if** (**this**.checkedCartItems.**length**===0) **return   if** (**window**.confirm(**'确定要删除吗?'**)) {  **const** promises = **this**.checkedCartItems.map(item => {  **return this**.$store.**dispatch**(**'deleteCartItem'**, item.skuId)  })  **await** Promise.all(promises)  **this**.getCartList()  }  }  }  } </**script**> |

## 注册与登陆路由

### 重难点说明

1. 注册/登陆请求后组件的响应处理
2. 登陆后自动携带token数据

### 接口请求函数

|  |
| --- |
| *// 登陆 /api/user/passport/login* **export const** *reqLogin* = (mobile, password) => ajax.post(**'/user/passport/login'**, {mobile, password})  *// 退出登陆 /user/passport/logout* **export const** *reqLogout* = ()=>ajax.get(**'/user/passport/logout'**) *// 注册Register接口* **export const** *reqRegister* = (userInfo) => ajax.post(**'/user/passport/register'**, userInfo) |

### vuex

|  |
| --- |
| */\*  用于操作首页模块数据的vuex模块 \*/* **import** \* **as** storageUtils **from '@/utils/storageUtils' import** { reqLogin, reqLogout, reqRegister } **from '@/api'  const** state = {  **userInfo**: storageUtils.getUserInfo(), *// 登陆的用户信息对象* **userTempId**: storageUtils.getUserTempId(), *// 登陆前的唯一标识, 发送请求时需要携带* }  **const** mutations = {  RECEIVE\_USER\_INFO(state, userInfo) {  state.**userInfo** = userInfo  },   RESET\_USER\_INFO(state) {  state.**userInfo** = {}  }, }  **const** actions = {  */\*   登陆的异步action  \*/* **async** login({commit}, {mobile, password}) {  **const** result = **await** reqLogin(mobile, password)  **if** (result.**code** === 200) {  **const** userInfo = result.**data** storageUtils.saveUserInfo(userInfo)  commit(**'RECEIVE\_USER\_INFO'**, userInfo)  **return ''** } **else** {  **return '登陆失败'** }  },   */\*   注册的异步action  \*/* **async** register ({commit}, {callback, ...userParams}) {  **const** result = **await** reqRegister(userParams)  **if** (result.**code** === 200) {  callback(**''**)  } **else** {  callback(result.**data** || **'注册失败, 请重新注册'**)  }  },   */\*   退出登陆的异步action  \*/* **async** logout({commit}) {  **const** result = **await** reqLogout()  **if** (result.**code** === 200) {  storageUtils.removeUserInfo()  commit(**'RESET\_USER\_INFO'**)  }  } }  **export default** {  state,  actions,  mutations } |

### 工具函数

|  |
| --- |
| */\*  localStorage存储管理的工具函数模块 \*/* **import** store **from 'store' import** UUID **from 'uuidjs'  const** USER\_INFO\_KEY = **'user\_info\_key' const** USER\_TEMP\_ID\_KEY = **'user\_temp\_id\_key'  export function** *saveUserInfo* (user) {  store.set(USER\_INFO\_KEY, user) }  **export function** *getUserInfo*() {  **return** store.get(USER\_INFO\_KEY) || {} }  **export function** *removeUserInfo*() {  store.remove(USER\_INFO\_KEY) }  */\*  获取用户临时ID \*/* **export function** *getUserTempId* () {  *// 从localStorage中读取* **let** userTempId = store.get(USER\_TEMP\_ID\_KEY)  *// 如果localStorage中没有, 生成一个新的, 并保存到local中* **if** (!userTempId) {  userTempId = UUID.*generate*()  store.set(USER\_TEMP\_ID\_KEY, userTempId)  }  *// 返回它* **return** userTempId } |

### 注册路由组件

|  |
| --- |
| <**template**>  <**div class="register"**>  <**h3**>注册新用户  <**span class="go"**>  我有账号，去 <**router-link to="/login"**>登陆</**router-link**>  </**span**>  </**h3**>  <**div class="content"**>  <**label**>手机号:</**label**>  <**input type="text" placeholder="请输入你的手机号" v-model="mobile"**>  </**div**>  <**div class="content"**>  <**label**>验证码:</**label**>  <**input type="text" placeholder="请输入验证码" v-model="*code*"**>  <**img ref="code" src="/api/user/passport/code" alt="code" @click="**updateCode**"**>  </**div**>  <**div class="content"**>  <**label**>登录密码:</**label**>  <**input type="text" placeholder="请输入你的登录密码" v-model="*password*"**>  </**div**>  <**div class="content"**>  <**label**>确认密码:</**label**>  <**input type="text" placeholder="请输入确认密码" v-model="password2"**>  </**div**>  <**div class="controls"**>  <**input name="m1" type="checkbox" v-model="agree"**>  <**span**>同意协议并注册《尚品汇用户协议》</**span**>  </**div**>  <**div class="btn" @click="**register**"**>  <**a href="javascript:"**>完成注册</**a**>  </**div**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **'Register'**,  data() {  **return** {  **mobile**: **''**,  **password**: **''**,  **password2**: **''**,  **code**: **''**,  **agree**: **true** };  },  **methods**: {  register() {  **const** {  mobile,  password,  password2,  code,  agree,  callback  } = **this** *// 表单验证* **this**.$store.**dispatch**(**'register'**, {  mobile,  password,  code,  callback  })  },  callback(errorMsg) {  **if** (!errorMsg) {  **this**.$router.replace({**path**: **'/login'**, **query**: **this**.$route.query})  } **else** {  *alert*(errorMsg)  }  },   updateCode() {  **this**.**$refs**.**code**.**src** = **`/api/user/passport/code?time=**${Date.now()}**`** }  }  } </**script**> |

### 登陆路由组件

|  |
| --- |
| <**template**>  <**div class="login-box"**>  *<!-- 登录 -->* <**div class="login-wrap"**>  <**div class="login"**>  <**div class="loginform"**>  <**ul class="tab clearFix"**>  <**li**>  <**a href="##" style="border-right: 0;"**>扫描登录</**a**>  </**li**>  <**li**>  <**a href="##" class="current"**>账户登录</**a**>  </**li**>  </**ul**>   <**div class="content"**>  <**form action="##"**>  <**div class="input-text clearFix"**>  <**span**></**span**>  <**input type="text" placeholder="手机号" v-model="mobile"**>  </**div**>  <**div class="input-text clearFix"**>  <**span class="pwd"**></**span**>  <**input type="password" placeholder="请输入密码" v-model="*password*"**>  </**div**>  <**div class="setting clearFix"**>  <**label class="checkbox inline"**>  <**input name="m1" type="checkbox"**>  自动登录  </**label**>  <**span class="forget"**>忘记密码？</**span**>  </**div**>  <**button class="btn" @click.prevent="**login**"**>登**&nbsp;&nbsp;**录</**button**>  </**form**>  <**div class="call clearFix"**>  <**ul**>  <**li**><**img src="./images/qq.png" alt=""**></**li**>  <**li**><**img src="./images/sina.png" alt=""**></**li**>  <**li**><**img src="./images/ali.png" alt=""**></**li**>  <**li**><**img src="./images/weixin.png" alt=""**></**li**>  </**ul**>  <**router-link class="register" :to="**{**path**: **'/register'**, **query**: $route.query}**"**>注册</**router-link**>  </**div**>  </**div**>  </**div**>  </**div**>  </**div**>  *<!-- 底部 -->* <**div class="copyright"**>  <**ul**>  <**li**>关于我们</**li**>  <**li**>联系我们</**li**>  <**li**>联系客服</**li**>  <**li**>商家入驻</**li**>  <**li**>营销中心</**li**>  <**li**>手机尚品汇</**li**>  <**li**>销售联盟</**li**>  <**li**>尚品汇社区</**li**>  </**ul**>  <**div class="address"**>地址：北京市昌平区宏福科技园综合楼6层</**div**>  <**div class="beian"**>京ICP备19006430号</**div**>  </**div**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **"Login"**,  data() {  **return** {  **mobile**: **"12311111111"**,  **password**: **"111111"** }  },  **methods**: {  **async** login() {  **const** {mobile,password,callback} = **this**;  **const** errorMsg = **await this**.$store.**dispatch**(**'login'**, {mobile, password})  **if** (!errorMsg) {  **const** redirect = **this**.$route.query.redirect  **this**.$router.replace(redirect || **'/'**)  } **else** {  *alert*(errorMsg)  }  },  }  } </**script**> |

### 登陆后请求自动携带token数据

|  |
| --- |
| */\* 6. 每次请求如果存在token, 携带到token请求头 \*/* **const** userInfo = store.**state**.**user**.*userInfo* **if** (userInfo && userInfo.**token**) {  config.headers[**'token'**] = userInfo.**token** } |

## 导航/路由守卫

### 重难点说明

1. 理解导航守卫
2. 使用导航守卫实现以下功能
   1. 只有登陆了, 才能查看交易/支付/个人中心界面
   2. 只有没有登陆, 才能查看登陆界面
   3. 只有携带的skuNum以及sessionStorage中有skuInfo数据, 才能查看添加购物车成功的界面
   4. 只能从购物车界面, 才能跳转到交易界面
   5. 只能从交易界面, 才能跳转到支付界面
   6. 只有从支付界面, 才能跳转到支付成功的界面

### 导航守卫

1. **导航守卫是什么?**

1). 导航守卫是vue-router提供的下面2个方面的功能

a. 监视路由跳转 -->回调函数

b. 控制路由跳转

2). 应用

a. 在跳转到界面前, 进行用户权限检查限制(如是否已登陆)

b. 在界面离开前, 做收尾工作

1. **导航守卫分类**

1). 全局守卫: 针对任意路由跳转

a. 全局前置守卫

b. 全局后置守卫

2). 路由独享的守卫

前置守卫

3). 组件守卫: 只针对当前组件的路由跳转

a. 进入

b. 更新

c. 离开

1. **相关API**

1). 全局前置守卫: 在准备跳转到某个路由组件之前 (在开发中用的比较多)

router.beforeEach((to, from, next) => {// before enter each route component

})

说明:

①. to: 目标route

②. from: 起始route

③. next: 函数

next(): 执行下一个守卫回调, 如果没有跳转到目标路由

next(false)/不执行: 跳转流程在当前处中断, 不会跳转到目标路由组件

next(path): 跳转到指定的另一个路由

2). 全局后置守卫: 在跳转到某个路由组件之后

router.afterEach((to, from) => {

})

3). 路由独享守卫

beforeEnter: (to, from, next) => {

}

3). 组件守卫

// 在当前组件对象被创建前调用, 不能直接访问this(不是组件对象)

// 但可以通过next(component => {}), 在回调函数中访问组件对象

beforeRouteEnter (to, from, next) {

next(component => {})

},

// 当前组件对象将要更新前调用, 可以访问this

beforeRouteUpdate (to, from, next) {

},

// 在当前组件离开前调用, 可以访问this

beforeRouteLeave (to, from, next) {

next()

}

### 使用导航守卫完成功能

1. **只有登陆了, 才能查看交易/支付/个人中心界面**

|  |
| --- |
| *// 所有需要检查登陆的路由路径的数组* **const** checkPaths = [**'/trade'**, **'/pay'**, **'/center'**]  */\*  注册全局前置拦截器 \*/* router.*beforeEach*((to, from, next) => {  **const** targetPath = to.**path  if** (checkPaths.some(path => targetPath.indexOf(path)===0)) {  **if** (!store.**state**.**user**.*userInfo*.**token**) {  **return** next(**`/login?redirect=**${targetPath}**`**)  }  }   next() }) |

1. **只有没有登陆, 才能查看登陆界面**

|  |
| --- |
| {  **path**: **'/login'**,  **component**: Login,  **meta**: {  **isHideFooter**: **true**, *// 标识footer是否隐藏* },  beforeEnter(to, from, next) {  **if** (store.**state**.**user**.*userInfo*.**token**) {*// 如果已登陆, 直播跳转到首页* next(**'/'**)  } **else** {  next()  }  } } |

1. **只有携带的skuNum以及sessionStorage中有skuInfo数据, 才能查看添加购物车成功的界面**

|  |
| --- |
| {  **path**: **'/addcartsuccess'**,  **component**: AddCartSuccess,  props: route => route.**query**,  beforeEnter: (to, from, next) => {  **const** {skuId, skuNum} = to.**query  const** skuInfo = ***JSON***.parse(**window**.**sessionStorage**.getItem(**'SKU\_INFO'**))  **if** (skuNum>0 && skuInfo && skuInfo.**id**) {  next()  } **else** {  next(**'/'**)  }  } }, |

1. **只能从购物车界面, 才能跳转到交易界面**

|  |
| --- |
| {  **path**: **'/trade'**,  **component**: Trade,  beforeEnter: (to, from, next) => {  **if** (from.**path**!==**'/shopcart'**) {  next({**path**: **'/shopcart'**})  } **else** {  next()  }  } } |

1. **只能从交易界面, 才能跳转到支付界面**

|  |
| --- |
| {  **path**: **'/pay'**,  **component**: Pay,  beforeEnter: (to, from, next) => {  **if** (from.**path**!==**'/trade'**) {  next({**path**: **'/trade'**})  } **else** {  next()  }  } }, |

1. **只有从支付界面, 才能跳转到支付成功的界面**

|  |
| --- |
| **export default** {  **name**: **'PaySuccess'**,   beforeRouteEnter: (to, **from**, next) => {  **if** (**from**.**path**!==**'/pay'**) {  next({**path**: **'/pay'**})  } **else** {  next()  }  } } |

## 订单与支付

### 重难点说明

1. 提交订单
2. 支付二维码
3. 获取订单状态

### 下载依赖

npm install -S qrcode element-ui

npm install -D babel-plugin-component

### UI组件库的按需引入打包

1. babel.config.js

|  |
| --- |
| **"plugins"**: [  [  **"component"**,  {  **"libraryName"**: **"element-ui"**,  **"styleLibraryName"**: **"theme-chalk"** }  ] ] |

1. src/elements.js

|  |
| --- |
| **import** Vue **from 'vue' import** {  MessageBox,  Message,  Pagination } **from 'element-ui'** Vue.component(Pagination.**name**, Pagination);  Vue.**prototype**.**$MessageBox** = MessageBox Vue.**prototype**.$alert = MessageBox.alert Vue.**prototype**.$message = Message |

1. 入口JS中加载

|  |
| --- |
| **import './elements'** *// 加载element-ui需要的组件* |

### 接口请求函数

|  |
| --- |
| *// 获取订单交易页信息* **export const** *reqTradeInfo* = () => ajax.get(**'/order/auth/trade'**) *// 提单订单* **export const** *reqSubmitOrder* = (tradeNo, orderInfo) => ajax({  **method**: **'POST'**,  **url**: **'/order/auth/submitOrder'**,  **params**: {tradeNo},  **data**: orderInfo }) *// 获取订单支付信息* **export const** *reqPayInfo* = (orderId) => ajax.get(**`/payment/weixin/createNative/**${orderId}**`**)  *// 查询支付订单状态* **export const** *reqOrderStatus* = (orderId) => ajax.get(**`/payment/weixin/queryPayStatus/**${orderId}**`**) *// 获取我的订单列表* **export const** *reqMyOrders* = (page, limit) => ajax.get(**`/order/auth/**${page}**/**${limit}**`**) |

### Vuex

|  |
| --- |
| */\* 订单的交易和支付数据管理 \*/* **import** { reqTradeInfo, reqPayInfo } **from '@/api'  const** state = {  **tradeInfo**: {}, *// 订单交易信息对象* **payInfo**: {} *// 支付信息对象* }  **const** mutations = {  RECEIVE\_TRADE\_INFO (state, {tradeInfo}) {  state.**tradeInfo** = tradeInfo  },  RECEIVE\_PAY\_INFO (state, {payInfo}) {  state.**payInfo** = payInfo  } }  **const** actions = {  */\*   获取订单交易信息的异步action  \*/* **async** getTradeInfo ({commit}) {  **const** result = **await** reqTradeInfo()  **if** (result.**code**===200) {  **const** tradeInfo = result.**data** commit(**'RECEIVE\_TRADE\_INFO'**, {tradeInfo})  }  },   */\*   获取支付信息的异步action  \*/* **async** getPayInfo ({commit}, orderId) {  **const** result = **await** reqPayInfo(orderId)  **if** (result.**code**===200) {  **const** payInfo = result.**data** commit(**'RECEIVE\_PAY\_INFO'**, {payInfo})  }  } }  **const** getters = {  */\*   需要支付的总金额  \*/* totalAmount (state) {  **return** state.**tradeInfo**.totalAmount  } }  **export default** {  state,  mutations,  actions,  getters } |

### 交易路由组件: Trade/index.vue

|  |
| --- |
| <**template**>  <**div class="trade-container"**>  <**h3 class="title"**>填写并核对订单信息</**h3**>  <**div class="content"**>  <**h5 class="receive"**>收件人信息</**h5**>  <**div class="address clearFix" v-for="**(addr, ***index***) in **tradeInfo**.userAddressList**"  :key="**addr.**id"** >  <**span class="username" :class="**{**selected**: **selectedAddr**===addr}**"**>{{addr.**consignee**}}</**span**>  <**p @click="selectedAddr**=addr**"**>  <**span class="s1"**>{{addr.userAddress}}</**span**>  <**span class="s2"**>{{addr.phoneNum}}</**span**>  <**span class="s3" v-if="**addr.isDefault===**'1'"**>默认地址</**span**>  </**p**>  </**div**>    <**div class="line"**></**div**>  <**h5 class="pay"**>支付方式</**h5**>  <**div class="address clearFix"**>  <**span class="username selected"**>在线支付</**span**>  <**span class="username" style="margin-left:5px;"**>货到付款</**span**>  </**div**>  <**div class="line"**></**div**>  <**h5 class="pay"**>送货清单</**h5**>  <**div class="way"**>  <**h5**>配送方式</**h5**>  <**div class="info clearFix"**>  <**span class="s1"**>天天快递</**span**>  <**p**>配送时间：预计8月10日（周三）09:00-15:00送达</**p**>  </**div**>  </**div**>   <**div class="line"**></**div**>  <**div class="detail"**>  <**h5**>商品清单</**h5**>  <**ul class="list clearFix" v-for="**(***item***, ***index***) in **tradeInfo**.detailArrayList**" :key="*item***.**skuId"**>  <**li**>  <**img :src="*item***.**imgUrl" alt="img"**>  </**li**>  <**li**>  <**p**>{{***item***.skuName}}</**p**>  <**h4**>7天无理由退货</**h4**>  </**li**>  <**li**>  <**h3**>￥{{***item***.orderPrice}}</**h3**>  </**li**>  <**li**>X{{***item***.**skuNum**}}</**li**>  <**li**>有货</**li**>  </**ul**>    </**div**>  <**div class="bbs"**>  <**h5**>买家留言：</**h5**>  <**textarea placeholder="建议留言前先与商家沟通确认" class="remarks-cont" v-model="orderComment"**></**textarea**>   </**div**>  <**div class="line"**></**div**>  <**div class="bill"**>  <**h5**>发票信息：</**h5**>  <**div**>普通发票（电子） 个人 明细</**div**>  <**h5**>使用优惠/抵用</**h5**>  </**div**>  </**div**>  <**div class="money clearFix"**>  <**ul**>  <**li**>  <**b**><**i**>{{**tradeInfo**.totalNum}}</**i**>件商品，总商品金额</**b**>  <**span**>¥{{**tradeInfo**.totalAmount}}</**span**>  </**li**>  <**li**>  <**b**>返现：</**b**>  <**span**>0.00</**span**>  </**li**>  <**li**>  <**b**>运费：</**b**>  <**span**>0.00</**span**>  </**li**>  </**ul**>  </**div**>  <**div class="trade"**>  <**div class="price"**>应付金额:　<**span**>¥{{**tradeInfo**.totalAmount}}</**span**></**div**>  <**div class="receiveInfo"**>  寄送至:  <**span**>{{**selectedAddr**.userAddress}}</**span**>  收货人：<**span**>{{**selectedAddr**.**consignee**}}</**span**>  <**span**>{{**selectedAddr**.phoneNum}}</**span**>  </**div**>  </**div**>  <**div class="sub clearFix"**>  <**a href="javascript:" class="subBtn" @click="**toPay**"**>提交订单</**a**>  </**div**>  </**div**> </**template**>  <**script**>  **import** { mapState } **from 'vuex'  export default** {  **name**: **'Trade'**,   data () {  **return** {  **selectedAddr**: {},  **orderComment**: **'能不能今天发货!'** }  },   **computed**: {  ...mapState({  tradeInfo: state => state.**order**.**tradeInfo** })  },   mounted () {  **this**.$store.**dispatch**(**'getTradeInfo'**)  },   **watch**: {  **'tradeInfo.userAddressList'** (value) {  **this**.**selectedAddr** = value.find(item => item.isDefault===**'1'**) || {}  }  },   **methods**: {  **async** toPay () {  */\*   {  "consignee": "admin",  "consigneeTel": "15011111111",  "deliveryAddress": "北京市昌平区2",  "paymentWay": "ONLINE",  "orderComment": "xxx",  "orderDetailList": [  {  "id": null,  "orderId": null,  "skuId": 6,  "skuName": " Apple iPhone 11 (A2223) 128GB 红色 移动联通电信22",  "imgUrl": "http://182.92.128.115:8080//rBFUDF6V0JmAG9XGAAGL4LZv5fQ163.png",  "orderPrice": 4343,  "skuNum": 2,  "hasStock": null  },  {  "id": null,  "orderId": null,  "skuId": 4,  "skuName": "Apple iPhone 11 (A2223) 128GB 红色",  "imgUrl": "http://182.92.128.115:80800/rBFUDF6VzaeANzIOAAL1X4gVWEE035.png",  "orderPrice": 5999,  "skuNum": 1,  "hasStock": null  }  ]  }  \*/  // 准备请求需要的数据  // 准备交易ID* **const** {tradeNo} = **this**.**tradeInfo** *// 准备订单相关信息对象* **const** orderInfo = {  **consignee**: **this**.**selectedAddr**.**consignee**,  **consigneeTel**: **this**.**selectedAddr**.phoneNum,  **deliveryAddress**: **this**.**selectedAddr**.userAddress,  **paymentWay**: **"ONLINE"**,  **orderComment**: **this**.**orderComment**,  **orderDetailList**: **this**.**tradeInfo**.detailArrayList  }   *// 发送提交订单的ajax请求* **const** result = **await this**.**$API**.reqSubmitOrder(tradeNo, orderInfo)  **if** (result.**code**===200) { *// 如果成功了, 跳转去支付  // 得到返回的订单ID* **const** orderId = result.**data  this**.$router.push({**path**: **'/pay'**, **query**: {orderId}})  } **else** { *// 如果失败了, 提示  alert*(result.**message** || **'提交生成订单失败!'**)  }    }  }  } </**script**> |

### 支付路由组件: Pay/index.vue

|  |
| --- |
| <**template**>  <**div class="pay-main"**>  <**div class="pay-container"**>  <**div class="checkout-tit"**>  <**h4 class="tit-txt"**>  <**span class="success-icon"**></**span**>  <**span class="success-info"**>订单提交成功，请您及时付款，以便尽快为您发货~~</**span**>  </**h4**>  <**div class="paymark"**>  <**span class="fl"**>请您在提交订单<**em class="orange time"**>4小时</**em**>之内完成支付，超时订单会自动取消。订单号：<**em**>145687</**em**></**span**>  <**span class="fr"**><**em class="lead"**>应付金额：</**em**><**em class="orange money"**>￥{{totalAmount}}</**em**></**span**>  </**div**>  </**div**>  <**div class="checkout-info"**>  <**h4**>重要说明：</**h4**>  <**ol**>  <**li**>尚品汇商城支付平台目前支持<**span class="zfb"**>支付宝</**span**>支付方式。</**li**>  <**li**>其它支付渠道正在调试中，敬请期待。</**li**>  <**li**>为了保证您的购物支付流程顺利完成，请保存以下支付宝信息。</**li**>  </**ol**>  <**h4**>支付宝账户信息：（很重要，<**span class="save"**>请保存！！！</**span**>）</**h4**>  <**ul**>  <**li**>支付帐号：11111111</**li**>  <**li**>密码：111111</**li**>  <**li**>支付密码：111111</**li**>  </**ul**>  </**div**>  <**div class="checkout-steps"**>  <**div class="step-tit"**>  <**h5**>支付平台</**h5**>  </**div**>  <**div class="step-cont"**>  <**ul class="payType"**>  <**li**><**img src="./images/pay2.jpg"**></**li**>  <**li**><**img src="./images/pay3.jpg"**></**li**>  </**ul**>   </**div**>  <**div class="hr"**></**div**>   <**div class="payshipInfo"**>  <**div class="step-tit"**>  <**h5**>支付网银</**h5**>  </**div**>  <**div class="step-cont"**>  <**ul class="payType"**>  <**li**><**img src="./images/pay10.jpg"**></**li**>  <**li**><**img src="./images/pay11.jpg"**></**li**>  <**li**><**img src="./images/pay12.jpg"**></**li**>  <**li**><**img src="./images/pay13.jpg"**></**li**>  <**li**><**img src="./images/pay14.jpg"**></**li**>  <**li**><**img src="./images/pay15.jpg"**></**li**>  <**li**><**img src="./images/pay16.jpg"**></**li**>  <**li**><**img src="./images/pay17.jpg"**></**li**>  <**li**><**img src="./images/pay18.jpg"**></**li**>  <**li**><**img src="./images/pay19.jpg"**></**li**>  <**li**><**img src="./images/pay20.jpg"**></**li**>  <**li**><**img src="./images/pay21.jpg"**></**li**>  <**li**><**img src="./images/pay22.jpg"**></**li**>   </**ul**>  </**div**>   </**div**>  <**div class="hr"**></**div**>   <**div class="submit"**>  <**a href="javascript:" class="btn" @click="**pay**"**>立即支付</**a**>  </**div**>  <**div class="otherpay"**>  <**div class="step-tit"**>  <**h5**>其他支付方式</**h5**>  </**div**>  <**div class="step-cont"**>  <**span**><**a href="weixinpay.html" target="\_blank"**>微信支付</**a**></**span**>  <**span**>中国银联</**span**>  </**div**>  </**div**>  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **import** { mapGetters, mapState } **from 'vuex'  import** QRCode **from 'qrcode'  export default** {  **name**: **'Pay'**,   **props**: {  **orderId**: ***Number*** },   mounted () {  **this**.$store.**dispatch**(**'getPayInfo'**, **this**.**orderId**)  },   **computed**: {  ...mapGetters([**'totalAmount'**]),  ...mapState({  payInfo: state => state.**order**.**payInfo** })  },   **methods**: {  */\*   生成支付二维码界面  \*/* **async** pay () {  *// 根据支付url生成二维码图片url* **const** url = **await** QRCode.toDataURL(**this**.**payInfo**.codeUrl)  *// 显示二维码图片提示* **this**.$alert(**`<img src="**${url}**" />`**, **'请使用微信扫码支付'**, {  **dangerouslyUseHTMLString**: **true**,  **showCancelButton**: **true**,  **confirmButtonText**: **'我已成功支付'**,  **cancelButtonText**: **'支付中遇到了问题'**,  **center**: **true**,  **showClose**: **false** })  .then(() => { *// 如果点击了已支付, 跳转到支付成功的界面* **this**.$router.replace(**'/paysuccess'**)  })  .catch(() => { *// 如果点击了有问题, 提示  clearInterval*(**this**.**intervalId**)  **this**.**intervalId** = 0  **this**.$message.warning(**'请联系尚硅谷漂亮的前台'**)  })   *// 如果没有正在运行的循环定时器* **if** (!**this**.**intervalId**) {  *// 启动3s的循环定时器, 获取订单的状态* **this**.**intervalId** = *setInterval*(() => {  *// console.log('setInterval.....')  // 异步获取订单的状态* **this**.**$API**.reqOrderStatus(**this**.**orderId**)  .then(result => {  ***console***.log(**'支付结果'**, result)  **if** (result.**code** === 200) { *// 如果支付成功了, 跳转到支付成功的路由去* **this**.$router.replace(**'/paysuccess'**)  }  })  .catch(error => { *// 如果失败了, 提示  clearInterval*(**this**.**intervalId**)  **this**.**intervalId** = 0  **this**.$message.error(**'支付未出错,请确认微信付款情况'**)  })  }, 3000)  }  }  },   beforeDestroy() {  ***console***.dir(**this**.**$MessageBox**)  **if** (typeof **this**.**intervalId**===0) {  **this**.**$MessageBox**.close() *// 关闭提示框* }  **if** (**this**.**intervalId**>0) { *// 如果定时器还没有清除, 清除定时器  clearInterval*(**this**.**intervalId**)  **this**.**intervalId** = 0  }  }  } </**script**> |

### 我的订单路由组件: Center/MyOrder/index.vue

|  |
| --- |
| <**template**>  <**div class="order-content"**>  <**div class="title"**>  <**h3**>我的订单</**h3**>  </**div**>  <**div class="chosetype"**>  <**table**>  <**thead**>  <**tr**>  <**th width="29%"**>商品</**th**>  <**th width="31%"**>订单详情</**th**>  <**th width="13%"**>收货人</**th**>  <**th**>金额</**th**>  <**th**>状态</**th**>  <**th**>操作</**th**>  </**tr**>  </**thead**>  </**table**>  </**div**>  <**div class="orders"**>   <**table class="order-item" v-for="**(***order***, ***index***) in **orders" :key="*order***.**id"**>  <**thead**>  <**tr**>  <**th colspan="5"**>  <**span class="ordertitle"**>  {{***order***.createTime}}　订单编号：{{***order***.outTradeNo}}  <**span class="pull-right delete"**>  <**img src="./images/delete.png"**>  </**span**>  </**span**>  </**th**>  </**tr**>  </**thead**>  <**tbody**>  <**tr v-for="**(***item***, ***index***) in ***order***.**orderDetailList" :key="*item***.**id"**>  <**td width="60%"**>  <**div class="typographic"**>  <**img :src="*item***.**imgUrl"**>  <**a href="#" class="block-text"**>{{***item***.skuName}}</**a**>  <**span**>x{{***item***.**skuNum**}}</**span**>  <**a href="#" class="service"**>{{***item***.orderPrice}}元</**a**>  </**div**>  </**td**>   <**template v-if="*index***===0**"**>  <**td rowspan="2" width="8%" class="center"**>{{***order***.**consignee**}}</**td**>  <**td rowspan="2" width="13%" class="center"**>  <**ul class="unstyled"**>  <**li**>总金额¥{{***order***.totalAmount}}</**li**>  <**li**>{{***order***.**paymentWay**==='ONLINE' ? '在线支付' : '货到付款'}}</**li**>   </**ul**>  </**td**>  <**td rowspan="2" width="8%" class="center"**>  <**a href="javascript:" class="btn"**>{{***order***.orderStatusName}}</**a**>  </**td**>  <**td rowspan="2" width="13%" class="center"**>  <**ul class="unstyled"**>  <**li**>  <**a href="javascript:"**>评价|晒单</**a**>  </**li**>  </**ul**>  </**td**>  </**template**>  </**tr**>  </**tbody**>  </**table**>   </**div**>  <**Pagination   :page-config="**{  ***total***,  **pageSize**: **limit**,  **pageNo**,  **showPageNo**: 5,  }**"  @changeCurrentPage="**getMyOrders**"** />  </**div**> </**template**>  <**script**>  **export default** {  **name**: **'MyOrder'**,   data () {  **return** {  **orders**: [],  **total**: 1,  **pageNo**: 1,  **limit**: 3  }  },   mounted () {  **this**.getMyOrders()  },   **methods**: {  */\*   异步获取指定页码的订单分页列表  \*/* **async** getMyOrders (pageNo=1) {  **this**.**pageNo** = pageNo  **const** result = **await this**.**$API**.reqMyOrders(**this**.**pageNo**, **this**.**limit**)  ***console***.log(result)  **if** (result.**code**===200) {  **const** {total, records} = result.**data  this**.**orders** = records  **this**.**total** = total  }  }  }  } </**script**> |

## 其它

### 图片懒加载

1. **图片懒加载特点说明**
   1. 还没有加载得到目标图片时, 先显示loading图片
   2. 在<img>进入可视范围才加载请求目标图片
2. **下载依赖**

npm install vue-lazyload

1. **引入并配置loading图片**

|  |
| --- |
| **import** VueLazyload **from 'vue-lazyload'**  **import** loading **from '@/assets/images/loading.gif'**  *// 在图片界面没有进入到可视范围前不加载, 在没有得到图片前先显示loading图片* Vue.use(VueLazyload, { *// 内部自定义了一个指令lazy* loading, *// 指定未加载得到图片之前的loading图片* }) |

1. **对异步获取的图片实现懒加载**

|  |
| --- |
| <**img v-lazy="**goods.defaultImg**"** /> |

### 路由懒加载

1. **理解**
   1. 当打包构建应用时，JS包会变得非常大，影响页面加载。如果我们能**把不同路由对应的组件分割成不同的代码块**，然后当路由被访问的时候才加载对应组件，这样就更加高效了
   2. 本质就是**Vue 的异步组件**在路由组件上的应用
   3. 需要使用动态import语法, 也就是**import()函数**
2. 编码

|  |
| --- |
| *// import Home from '@/pages/Home' // import Search from '@/pages/Search' // import Detail from '@/pages/Detail'  /\*  1. import('模块路径'): webpack会对被引入的模块单独打包 2. 路由函数只在第一次请求时才执行, 也就是第一次请求访问对应路由路径时才会请求后台加载对应的js打包文件 \*/* **const** *Home* = () => **import**(**'@/pages/Home'**) **const** *Search* = () => **import**(**'@/pages/Search'**) **const** *Detail* = () => **import**(**'@/pages/Detail'**) |

### 前台表单校验

1. **说明**
   1. 项目中有一些如注册/登陆表单, 在提交请求前是需要进行表单输入数据校验的
   2. 只有前台表单验证成功才会发请求
   3. 如果校验失败, 以界面红色文本的形式提示, 而不是用alert的形式
   4. 校验的时机, 除了点击提交时, 还有输入过程中实时进行校验
2. **下载依赖**

npm install vee-validate

1. **编码**
   1. **src/validate.js**

|  |
| --- |
| */\*  使用vee-validate进行表单验证 \*/* **import** Vue **from 'vue' import** { *extend*, ValidationObserver, ValidationProvider, *localize*, configure } **from 'vee-validate' import** \* **as** rules **from 'vee-validate/dist/rules' import** zh\_CN **from 'vee-validate/dist/locale/zh\_CN.json' import** merge **from 'lodash/merge'** *// 用于合并多个对象  // 配置错误提示文本的样式类名* configure({  **classes**: {  **valid**: **'is-valid'**, *// 验证合法的类名* **invalid**: **'is-invalid'**, *// 验证不合法的类名* } })  *// 注册所有规则* Object.keys(rules).forEach(rule => {  *extend*(rule, rules[rule]) })  *// 将自定义的message与内置的中文message合并* **const** locale = merge(zh\_CN, {  **messages**: {   **is**: **'{\_field\_}必须与密码相同'**,  **oneOf**: **'{\_field\_}必须同意'** } }) *// 指定中文提示信息 localize*(**'zh\_CN'**, locale)  *// 注册用于校验的组件* Vue.component(**'ValidationProvider'**, ValidationProvider) *// 用于输入过程中实时校验* Vue.component(**'ValidationObserver'**, ValidationObserver) *// 用于点击按钮时统一校验  /\*  https://github.com/logaretm/vee-validate 基本使用 https://logaretm.github.io/vee-validate/guide/rules.html#importing-the-rules 注册所有校验规则 https://logaretm.github.io/vee-validate/guide/state.html#css-classes 校验失败的样式类名 https://logaretm.github.io/vee-validate/guide/forms.html#basic-example 提交表单时统一校验 https://logaretm.github.io/vee-validate/guide/localization.html 指定本地(中文)提示信息 \*/* |

* 1. **main.js**

|  |
| --- |
| **import './validate'** *// 引入表单验证* |

* 1. **Register组件**

|  |
| --- |
| <**template**>  <**div class="register-container"**>  *<!-- 注册内容 -->* <**div class="register"**>  <**h3**>注册新用户  <**span class="go"**>我有账号，去 <**router-link to="/login"**>登陆</**router-link**>  </**span**>  </**h3**>   <**ValidationObserver ref="form"**>  <**form**>  <**div class="content"**>  <**label**>手机号:</**label**>  <**ValidationProvider name="手机号" :rules="**{**required**: **true**, **regex**: /^1\d{10}$/}**"**>  <**template slot-scope="{errors, classes}"**>  <**input type="text" placeholder="请输入你的手机号" v-model="mobile" :class="*classes*"**>  <**span class="error-msg"**>{{errors[0]}}</**span**>  </**template**>  </**ValidationProvider**>  </**div**>    <**div class="content"**>  <**label**>验证码:</**label**>   <**ValidationProvider name="验证码" :rules="**{**required**: **true**, **regex**: /^.{4}$/}**"**>  <**template slot-scope="{errors, classes}"**>  <**input type="text" placeholder="请输入验证码" v-model="*code*" :class="*classes*"**>  *<!-- http://182.92.128.115 -->* <**img ref="code" src="/api/user/passport/code" alt="code" @click="**updateCode**"**>  <**span class="error-msg"**>{{ errors[0] }}</**span**>  </**template**>  </**ValidationProvider**>   </**div**>  <**div class="content"**>  <**label**>登录密码:</**label**>  <**ValidationProvider name="密码" :rules="**{**required**: **true**, **min**: 6, **max**: 10}**"**>  <**template slot-scope="{ errors, classes }"**>  <**input type="password" placeholder="请输入你的登录密码" v-model="*password*" :class="*classes*"**>  <**span class="error-msg"**>{{ errors[0] }}</**span**>  </**template**>  </**ValidationProvider**>  </**div**>  <**div class="content"**>  <**label**>确认密码:</**label**>  <**ValidationProvider name="确认密码" :rules="**{**required**: **true**, **is**: ***password***}**"**>  <**template slot-scope="{ errors,classes}"**>  <**input type="password" placeholder="请输入确认密码" v-model="password2" :class="*classes*"**>  <**span class="error-msg"**>{{ errors[0] }}</**span**>  </**template**>  </**ValidationProvider**>  </**div**>  <**div class="controls"**>  <**ValidationProvider name="协议" :rules="**{**oneOf**: [**true**]}**"**>  <**template slot-scope="{ errors,classes}"**>  <**input name="m1" type="checkbox" v-model="isAgree" :class="*classes*"**>  <**span**>同意协议并注册《尚品汇用户协议》</**span**>  <**span class="error-msg"**>{{errors[0]}}</**span**>  </**template**>  </**ValidationProvider**>  </**div**>   <**div class="btn"**>  <**button @click.prevent="*register*"**>完成注册</**button**>  </**div**>  </**form**>  </**ValidationObserver**>    </**div**>   *<!-- 底部 -->* <**div class="copyright"**>  <**ul**>  <**li**>关于我们</**li**>  <**li**>联系我们</**li**>  <**li**>联系客服</**li**>  <**li**>商家入驻</**li**>  <**li**>营销中心</**li**>  <**li**>手机尚品汇</**li**>  <**li**>销售联盟</**li**>  <**li**>尚品汇社区</**li**>  </**ul**>  <**div class="address"**>地址：北京市昌平区宏福科技园综合楼6层</**div**>  <**div class="beian"**>京ICP备19006430号  </**div**>  </**div**>  </**div**> </**template**>  <**script**>  **export default** {  **name**: **'Register'**,   data() {  **return** {  **mobile**: **''**, *// 手机号* **password**: **''**, *// 密码* **password2**: **''**, *// 确认密码* **code**: **''**, *// 一次性图形验证* **isAgree**: **false**, *// 是否同意* }  },   **methods**: {  register() {   **this**.$refs.**form**.validate().then(**async** (success) => {  **if** (!success) {  **return**;  }   **const** {  mobile,  password,  code,  } = **this** *// 2. 发送注册的请求* **try** {  **await this**.$store.**dispatch**(**'register'**, {  mobile,  password,  code  })  *// 3.1. 如果成功了, 跳转到登陆的界面* **this**.$router.replace(**'/login'**)  ***console***.log(**'注册成功'**)  } **catch** (error) {  *// 3.2. 如果失败了, 提示文本  alert*(error.**message**)  }  })    },   */\*   更新验证码显示: 告诉浏览器重新发请求获取验证码图片  \*/* updateCode() {  *// 给img重新指定src  // 如果src新的值与原本的值相同, 浏览器不会自动请求获取图片显示  // 解决: 携带时间(当前时间值)戳的参数 ==> 每次指定的src值都不一样==> 浏览器就会自动请求  // this.$refs.code.src = `http://182.92.128.115/api/user/passport/code` // 是一个http请求* **this**.$refs.**code**.**src** = **`/api/user/passport/code`** *// 是一个http请求* }  }  } </**script**> |