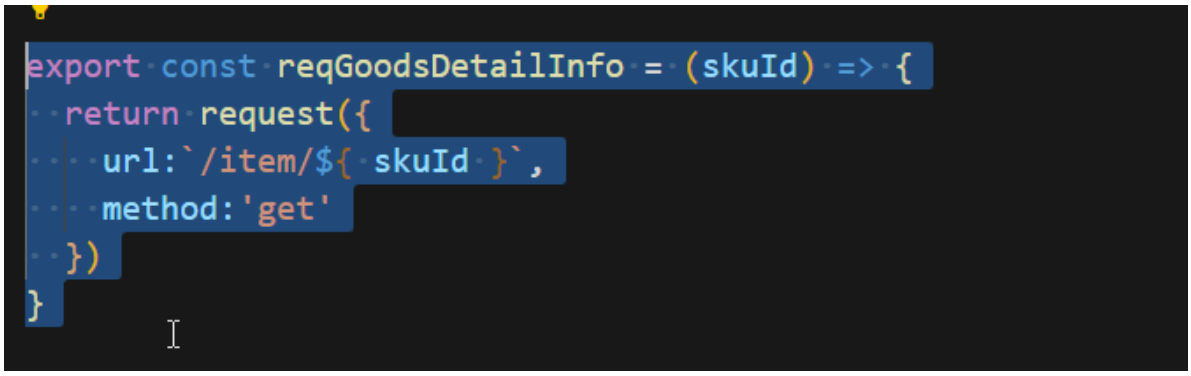
day07-前台项目07

逻辑思维

1. router-link to /detail + goods.id /detail + goods.title
2. 捞Detail静态
3. 配置router 路径注册 路由传参手机Id /:skuId
4. 配置路由的固定相 router index scrollBehavior 垂直滚动条初始位置
5. 初始化动态渲染

    i. api 接口请求函数 reqGoodsDetailInfo(skuId)

    a. 
```
export const reqGoodsDetailInfo = (skuId) => {
  return request({
    url:`/item/${ skuId }`,
    method:'get'
  })
}
```

    ii. store下新建小\store detail.js并合并 vux 三连环

    a. 
```
import { reqGoodsDetailInfo } from "@/api"

const state = {
  goodsDetailInfo:{}
  RECEIVE_GOODSDETA       (state,goodsDetailInfo){
    state.goodsDeta        = goodsDetailInfo
  }                AILINFO
}                  ailInfo
const actions = {
}
const actions = {
  async getGoodsDetailInfo({commit},skuId){
    try {
      const result = await reqGoodsDetailInfo(skuId)
      if(result.code === 200){
        commit('RECEIVE_GOODSDETAILINFO',result.data)
      }else{
        alert('获取商品详情失败')
      }
    } catch (error) {

    }
```

    iii. Detail组件里面 dispatch 发送请求

    a. mounter(){this.getGoodsDetailInfo()}

    b. methods:{getGoodsDetailInfo(){this.$store.dispatch('getGoodsDetailInfo', this.skuId)}}

c. beforeMount(){this.skuId = this.$route.params.skuId} 路由参数存data

iv. store里detail.js getters *** 注意初始化数据为 {} 或 []，不然state无数据undefined 因为 getters不等待，直接去拿 拿到undefined页面就直接报错

a.
```
categoryView(state){
  return state.goodsDetailInfo.categoryView || {}
},
skuInfo(state){
  return state.goodsDetailInfo.skuInfo || {}
},
spuSaleAttrList(state){
  return state.goodsDetailInfo.spuSaleAttrList || []
}
```

v. 数据捞至Detail 组件 computed ...mapGetters(['categoryView','skuInfo','spuSaleAttrList'])，然后Vue开发工具检查Detail组件数据是否拿到

6. 数据展示 面包屑 商品信息 销售属性
   i. 面包屑 留3个 对象没法遍历 直接写
   ii. 商品信息 InfoName 名字skuInfo.skuName 描述skuInfo.skuDesc 价格skuInfo.price
   iii. 销售属性 dl vfor saleAttr spuSaleAttrList spuSaleAttr.id
      a. saleAttr.saleAttrName
      b. vfor saleAttrValue saleAttrValueList
      c. saleAttrValue.id
      d. saleAttrValue.saleAttrValueName
7. 中图 小图 大图展示 公司UI三套图 同一套图组件通信 难度增加
   i. skuInfo skuImageList 7张图片 计算 传给中图和小图
      a. computed imageList return this.skuInfo.skuImageList || []
      b. Zoom ImagesList 组件传参
      c. ImageList props接收
      d. vfor image imageList
      e. Zoom props ['imageList']
      f. :src imageList[0].imgUrl undefined敢点敢错 假报错 计算加 || [] "currentImage.imgUrl"
      g. data return { currentIndex:0} computed:{currentImage return this.imageList[this.currentIndex] || {} } 动态数据 图片下标不能写死 为了防止undefined假报错，要去计算展示的图片
      h. big :src "currentImage.imgUrl"
8. 交互 销售属性 放大镜 小图 数量+- 加入购物车
   i. Detail组件 81行 : class="{active:saleAttrValue.isChecked === '1'}"

ii. 销售属性 排他 所有人一个状态 点击的一个状态
    a. @click changeIscheck
    b. methods changeIsCheck(spasaleAttrValueList,saleAttrValue) {spasaleAttrValueList.forEach(item => }
9. 小图 橙色框框 .active :class active:currentIndex @changeIndex
    i. 参考值 data currentIndex
    ii. methods changeIndex this.currentIndex = index
    iii. 小图传给中图 兄弟组件 全局事件总线 小图同步切换中图
    iv. thisbus.emit('changeIndex'，index)
    v. mounted(){this.$bus.$on('changIndex',this.changeIndex}
    vi. methods changeIndex(index){ this.currentIndex = index}
10. ImageList组件 Swiper sliderLoop 拷走watch 监视imageList
    i. swiper -contain ref this.$refs.imgSwiper
    ii. Swiper Carouser旋转木马 sliderPerView每屏显示几个图 sliderPerGroup几个小图一组滑动 loop干掉小 图轮播图over
11. 放大镜
    i. 放大镜的步骤：
        a. 鼠标动 遮罩层动
        b. 遮罩动 大图动
        c. 大图 动的 反向②倍
    ii. 组件Zoom
    iii. event @mousemove="move"
    iv. methods 放大镜鼠标移动事件绑定 move(event){}
        a. Zomm mask this.$refs.mask
        b. mouseX鼠标X的位置 event.offsetx
        c. mouseY鼠标Y的位置 exentoffsetY
        d. maskWidth mask.offsetWidth
        e. maskHeight mask.offsetHeight
        f. maskX mouseX - maskWidth / 2
        g. maskY mouseY - maskHeight / 2
        h. 判断边界 横向 纵向
        i. 横向边界 if maskX < 0 maskX = 0 maskX > maskWidth maskX = maskWidth
        j. 纵向边界 if maskY < 0 maskY = 0 maskY > maskHeight maskY = maskHeight
        k. mask.style.left = maskX + 'px'
        l. mask.style.top = maskY + 'px'
        m. bigImg this.$refs.bigImg
        n. bigImg.style.left

12. Detail组件 商品数量 v-model skuNum + @click skuNum+=1 - @click skuNum=skuNum <=1?1:skuNum-1
13. 失去焦点事件:blur事件 change改变事件 判断内容新值和老值是否一样 不一样触发
    i. @change skuNum = skuNum >= 1? parseInt(skuNum):1
14. 加入购物车 购物车是个数据 购物车数据添加后台数据库
    i. @click addShopCart methods addShopCart(){}
        a. 1、发请求 构造购物车数据 传到后台 后台存储
        b. 2、接收到后台返回响应，再根据响应结果判断是否跳转
        c. api reqAddOrUpdateShopCart= () =>{return request({url:`/cart/addToCart/${skuId/${skuNum}`,method:post})}
        d. vuex 三连环 shopCart 初始化并合并 actions
            a. addorUpdateShopCart({commit}),{skuId,skuNum}
            b. try...catch reqAddOrUpdateShopCart(skuId,skuNum)
            c. Detail组件 addShopCart(){this.$store.dispathch('addorUpdateShopCart',{skuId,skuNum})}
            d. if result.code 200 return "ok" else/catch return Promise.reject('failed')
            e. Detail组件 async addShopCart try...catch await this.$store.dispatch
            f.

```
async addShopCart(){
  let {skuId,skuNum} = this
  // 1、发请求，目的是把当前这个商品构造成购物车数据，传递到后台，后台需要把购物车存储到数据库
  // 存储之后才叫添加购物车成功，后台添加成功之后会返回给我们响应，
  try {
    //代表添加成功
    //相当于是在调用我们store当中的async函数，返回值一定是promise
    await this.$store.dispatch('addOrUpdateShopCart',{skuId,skuNum})
    alert('添加购物车成功，准备自动跳转到添加购物车成功页面')
  } catch (error) {
    //代表添加失败
    alert('添加购物车失败，留在原地')
  }

  // 2、接收到后台返回的响应之后，再根据响应的结果决定跳还是不跳
```

    ii. 捞添加到购物车成功 静态 src pages
        a. 配置 router
        b. Detail组件 addShopCart this.$router.push('/addcartsuccess')
        c. 简单数据 路由传参 无论params或query 复杂数据 别用路由传参采用存储方案解决
        d. 路由传参 '/addcartsucess?skuNum=' + skuNum
        e. 商品数量使用路由传参 商品的信息对象使用存储方案
        f. window.localStorage Web API
        g. sessionStorage.setItem('skuInfo_key',JSON.stingify(skuInfo))}
    iii. AddCartSuccesss组件
        a. return skuNum: '',skuInfo: {}
        b. beforeMounted
        c. 模板动态数据 :src skuInfo.skuDefaultImg skuInfo.skuName skuNum
        d. 辉洪 font捞入public font.css捞入css