

第五天

地址管理 + SKU + 购物车



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

地址模块 - 准备工作

新建分包页面



静态结构



动态设置标题

```
// 获取页面参数
const query = defineProps<{
  id?: string
}>()

// 动态设置标题
uni.setNavigationBarTitle({ title: query.id ? '修改地址' : '新建地址' })
```



地址模块 - 新建地址

封装API接口



定义参数类型



收集表单数据



点击保存调用



成功提示



返回上一页

```
/**
 * 添加收货地址
 * @param data 请求参数
 */
export const postMemberAddressAPI = (data: AddressParams) => {
  return http({
    method: 'POST',
    url: '/member/address',
    data,
  })
}
```

```
/** 添加收货地址: 请求参数 */
export type AddressParams = {
  /** 收货人姓名 */
  receiver: string
  /** 联系方式 */
  contact: string
  /** 省份编码 */
  provinceCode: string
  /** 城市编码 */
  cityCode: string
  /** 区/县编码 */
  countyCode: string
  /** 详细地址 */
  address: string
  /** 默认地址, 1为是, 0为否 */
  isDefault: number
}
```

```
// 表单数据
const form = ref({ ...省略 })

// 修改地区
const onRegionChange: UniHelper.RegionPickerOnChange = (ev) => {
  // 前端展示
  form.value.fullLocation = ev.detail.value.join(' ')
  // 后端参数
  const [provinceCode, cityCode, countyCode] = ev.detail.code!
  Object.assign(form.value, { provinceCode, cityCode, countyCode })
}

// 修改默认地址
const onSwitchChange: UniHelper.SwitchOnChange = (ev) => {
  form.value.isDefault = ev.detail.value ? 1 : 0
}

// 提交表单
const onSubmit = async () => {
  // 添加收货地址
  await postMemberAddressAPI(form.value)
  // 成功提示
  uni.showToast({ title: '添加成功' })
  // 返回上一页
  setTimeout(() => {
    uni.navigateBack()
  }, 400)
}
```



地址模块 - 列表渲染

封装API接口



初始化调用



定义类型(复用)



渲染列表

```
/**
 * 获取收货地址列表
 */
export const getMemberAddressAPI = () => {
  return http<AddressItem[]>({
    method: 'GET',
    url: '/member/address',
  })
}
```

```
/** 收货地址项 */
export type AddressItem = AddressParams & {
  /** 收货地址 id */
  id: string
  /** 省市区 */
  fullLocation: string
}
```

```
/** 商品详情信息 */
export type GoodsResult = {
  ...省略
  /** 用户地址列表[ 地址信息 ] */
  userAddresses: AddressItem[]
}
```

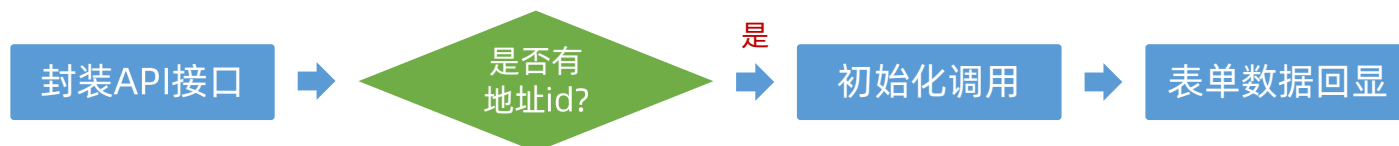
```
// 获取收货地址列表数据
const addressList = ref<AddressItem[]>([])
const getMemberAddressData = async () => {
  const res = await getMemberAddressAPI()
  addressList.value = res.result
}
// 初始化调用
onShow(() => {
  getMemberAddressData()
})
```

```
<view class="item" v-for="item in addressList" :key="item.id">
  <view class="item-content">
    <view class="user">
      {{ item.receiver }}
      <text class="contact">{{ item.contact }}</text>
      <text v-if="item.isDefault" class="badge">默认</text>
    </view>
    <view class="locate">{{ item.fullLocation }} {{ item.address }}</view>
    <navigator :url="`../address-form/address-form?id=${item.id}`">
      修改
    </navigator>
  </view>
</view>
```



注意事项：地址列表通过 **onShow** 初始化调用，因为新建地址成功后，需要**显示**最新收货地址列表。

地址模块 - 修改地址 - 数据回显



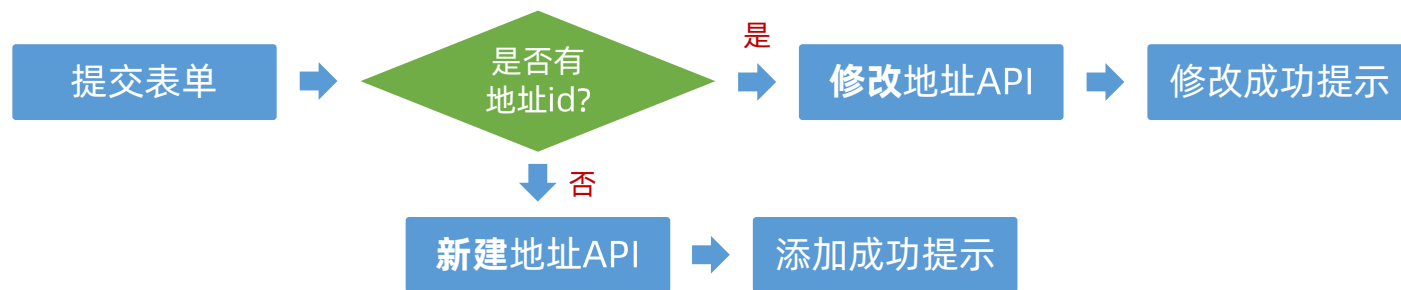
```
/**
 * 获取收货地址详情
 * @param id 地址id
 */
export const getMemberAddressByIdAPI = (id: string) => {
  return http<AddressItem>({
    method: 'GET',
    url: `/member/address/${id}`,
  })
}
```

```
// 根据id获取收货地址详情
const getMemberAddressByIdData = async () => {
  if (query.id) {
    // 发送请求
    const res = await getMemberAddressByIdAPI(query.id)
    // 合并到表单数据中
    Object.assign(form.value, res.result)
  }
}

// 页面加载时
onLoad(() => {
  getMemberAddressByIdData()
})
```

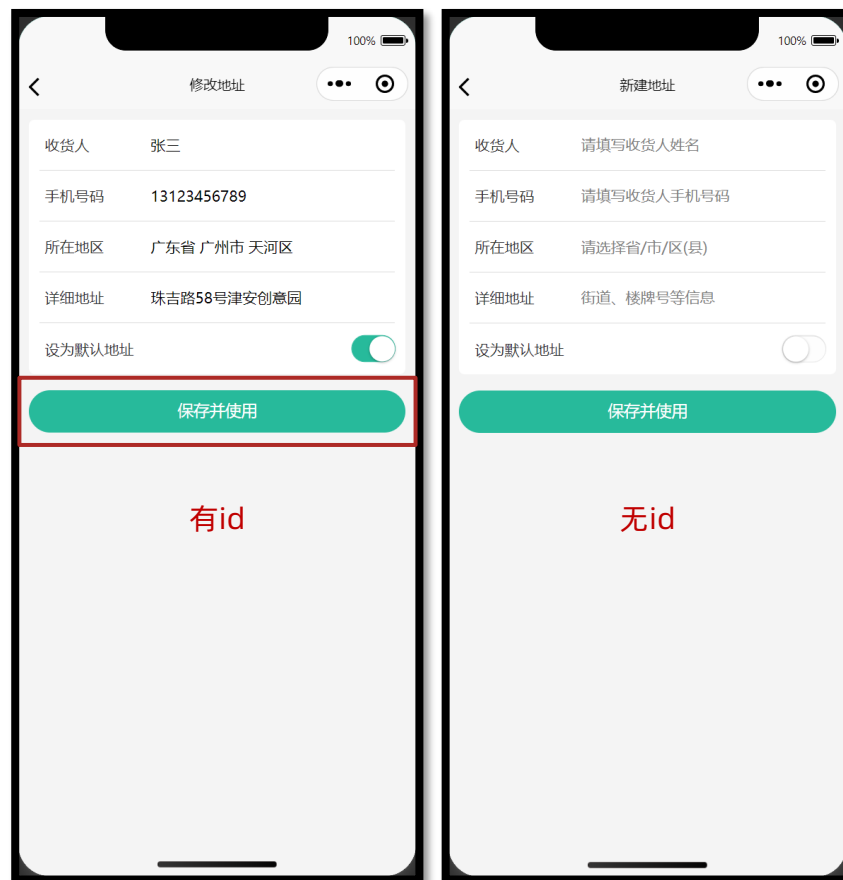


地址模块 - 修改地址 - 保存修改



```
/**
 * 修改收货地址
 */
export const putMemberAddressById = (id: string, data: AddressParams) => {
  return http({
    method: 'PUT',
    url: `/member/address/${id}`,
    data,
  })
}
```

```
// 提交表单
const onSubmit = async () => {
  if (query.id) {
    await putMemberAddressById(query.id, form.value) // 有id: 修改地址
  } else {
    await postMemberAddressAPI(form.value) // 无id: 新建地址
  }
  // 成功提示
  uni.showToast({ icon: 'success', title: query.id ? '修改成功' : '添加成功' })
  // ...省略
}
```



地址模块 - 表单校验

uni-forms 组件



定义校验规则



修改表单结构



绑定表单数据



提交时校验

[uni-form 组件文档](#)

```
// 表单校验规则
const rules: UniHelper.UniFormsRules = {
  receiver: {
    rules: [{ required: true, errorMessage: '请填写收货人' }],
  },
  contact: {
    rules: [
      { required: true, errorMessage: '请填写手机号码' },
      { pattern: /^1[3-9]\d{9}$/, errorMessage: '手机号码格式错误' },
    ],
  },
  ...省略
}
```



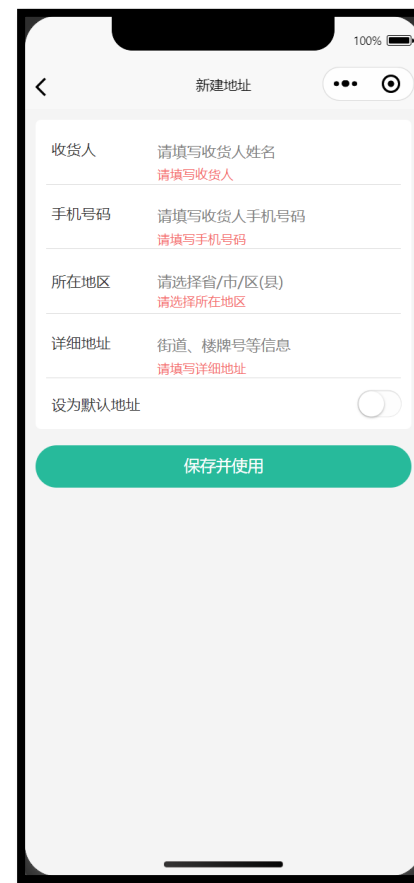
```
<uni-forms :rules="rules" :model="form" ref="formRef">
  <uni-forms-item name="receiver" class="form-item">
    <text class="label">收货人</text>
    <input placeholder="请填写收货人姓名" v-model="form.receiver" />
  </uni-forms-item>
  <uni-forms-item name="contact" class="form-item">
    <text class="label">手机号码</text>
    <input placeholder="请填写手机号码" v-model="form.contact" />
  </uni-forms-item>
  ...省略
</uni-forms>
```



```
// 表单数据
const form = ref({
  receiver: '', // 收货人
  contact: '', // 联系方式
  fullLocation: '', // 省市区(前端展示)
  provinceCode: '',
  cityCode: '',
  countyCode: '',
  address: '', // 详细地址
  isDefault: 0,
})
```



```
// 表单组件实例
const formRef = ref<UniFormsInstance>()
// 提交表单
const onSubmit = async () => {
  try {
    // 表单校验
    await formRef.value?.validate?.()
    // 校验通过, 发送请求
  } catch (error) {
    // 校验失败, 提示用户
  }
}
```



地址模块 - 删除地址

uni-swipe-action 组件



修改列表结构



绑定删除事件



二次确认删除



删除地址(API)

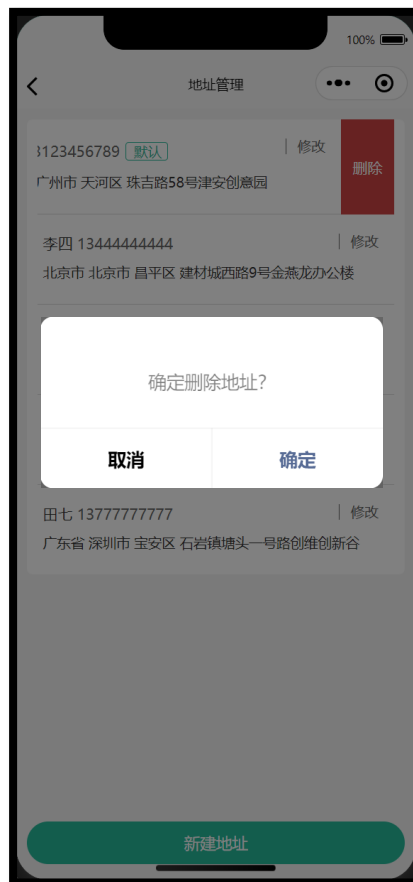
[uni-swipe-action 组件文档](#)

```
<uni-swipe-action> ----- 滑动操作分区
  <uni-swipe-action-item> ---- 滑动操作项
    <view>内容</view> ----- 默认插槽(放内容)
    <template #right> ----- 右侧插槽(放按钮)
      <button>删除</button>
    </template>
  </uni-swipe-action-item>
</uni-swipe-action>
```

```
<view v-if="addressList.length" class="address">
  <uni-swipe-action class="address-list">
    <uni-swipe-action-item class="item"
      v-for="item in addressList"
      :key="item.id">
      >
        <view class="item-content">
          ...省略
        </view>
        <template #right>
          <button @tap="onDeleteAddress(item.id)">
            删除
          </button>
        </template>
      </uni-swipe-action-item>
    </uni-swipe-action>
  </view>
```

```
// 删除收货地址
const onDeleteAddress = (id: string) => {
  // 弹窗二次确认
  uni.showModal({
    content: '确定删除地址?',
    async success(res) {
      if (res.confirm) {
        // 删除地址
        await deleteMemberAddressByIdAPI(id)
        // 获取最新地址列表
        getMemberAddressData()
      }
    },
  })
}
```

```
/**
 * 删除收货地址
 * @param id 地址id(路径参数)
 */
export const deleteMemberAddressByIdAPI = (id: string) => {
  return http({
    method: 'DELETE',
    url: `/member/address/${id}`,
  })
}
```



SKU模块 - 基本概念

SKU 概念：存货单位（Stock Keeping Unit），**库存**管理的最小可用单元，通常称为“单品”。

SKU 常见于电商领域，对于前端工程师而言，更多关注 **SKU 算法**，基于后端的 SKU 数据**渲染页面**并**实现交互**。

```
[ '黑色', '34' ]  
[ '黑色', '35' ]  
[ '黑色', '36' ]  
[ '黑色', '37' ]  
[ '黑色', '38' ]  
[ '黑色', '39' ]  
[ '黑色', '40' ]  
[ '杏色', '38' ]
```

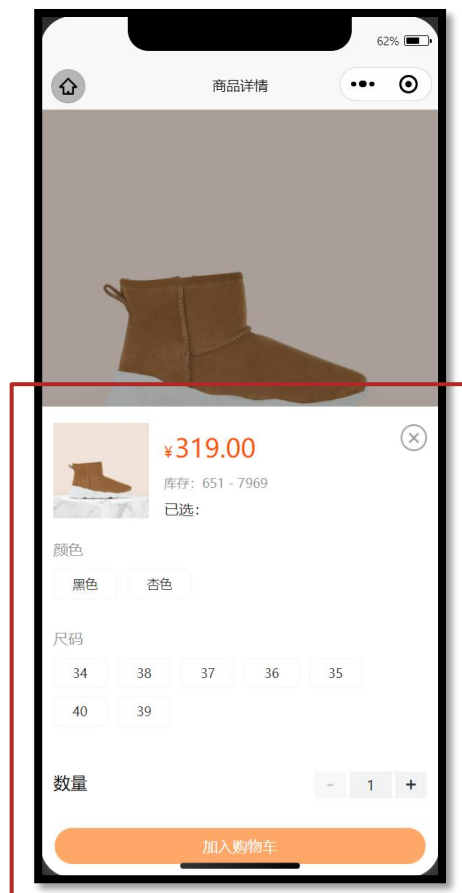
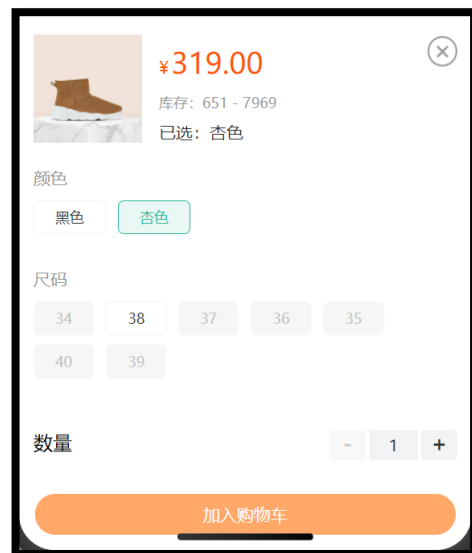
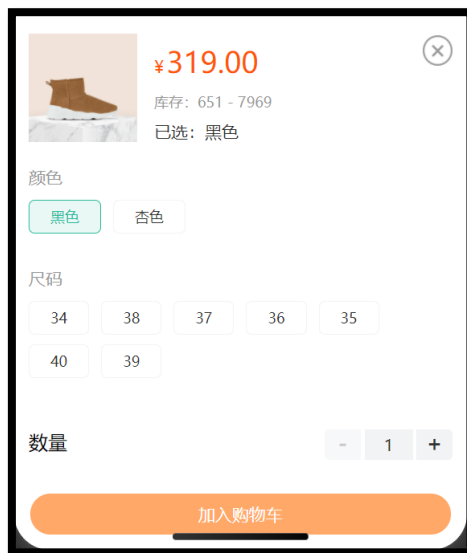
SKU算法



```
[ '黑色', '杏色' ]  
[ '34', '38', '37', '36', '35', '40', '39' ]
```

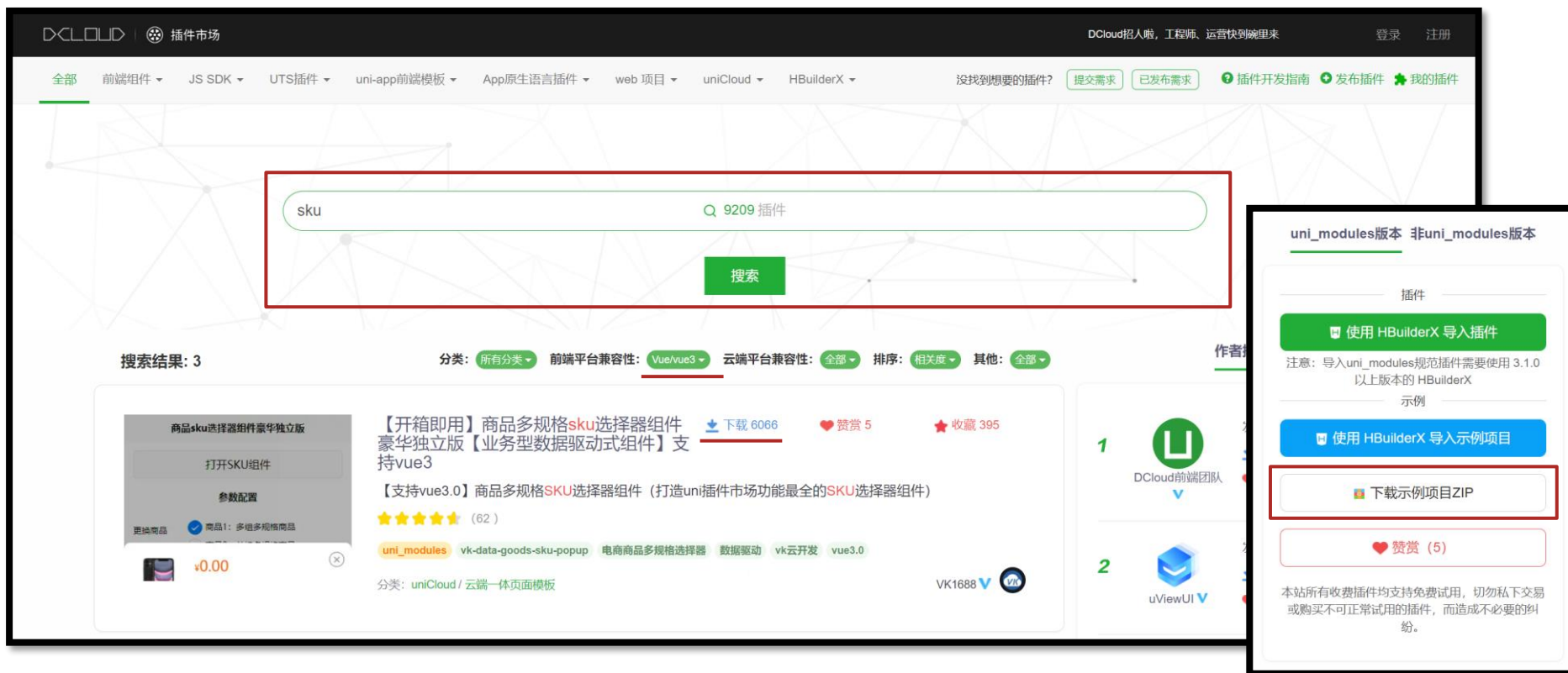


渲染页面



SKU模块 - 下载SKU插件

[uni-app插件市场](#)：是uni-app官方插件生态集中地，有数千款插件。



SKU模块 - 使用SKU插件

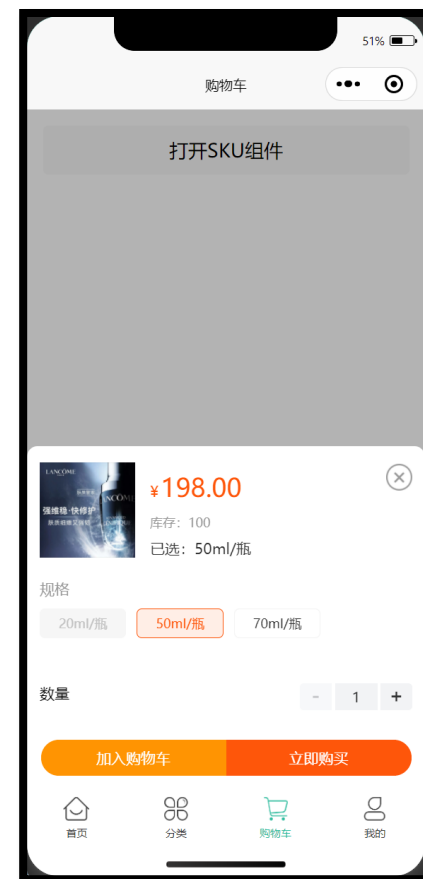
组件安装到自己项目

- 1、复制 **vk-data-goods-sku-popup** 和 **vk-data-input-number-box** 到项目的根 **components** 目录下。
- 2、复制例子代码并运行

注意事项：项目进行 git 提交时会校验文件，可添加 `/* eslint-disable */` 禁用检查。

```
<script>
/* eslint-disable */
export default {
  name: 'vk-data-goods-sku-popup',
  ...省略
}
</script>
```

```
<script>
/* eslint-disable */
export default {
  name: 'vk-data-input-number-box',
  ...省略
}
</script>
```



SKU模块 - 渲染商品信息

SKU弹窗组件



SKU组件文档



类型声明文件



显示SKU弹窗



渲染商品信息

```
<vk-data-goods-sku-popup  
  v-model="isShowSku"  
  :localdata="localdata"  
>
```

```
import { Component } from '@uni-helper/uni-app-types'  
  
/** SKU 弹出层 */  
export type SkuPopup = Component<SkuPopupProps>  
/** SKU 弹出层属性 */  
export type SkuPopupProps = {  
  /** 双向绑定, true 为打开组件, false 为关闭组件 */  
  modelValue: boolean  
  /** 商品信息本地数据源 */  
  localdata: SkuPopupLocaldata  
  ...省略  
}  
  
/** 全局组件类型声明 */  
declare module '@vue/runtime-core' {  
  export interface GlobalComponents {  
    'vk-data-goods-sku-popup': SkuPopup  
  }  
}
```

```
// SKU弹窗  
const isShowSku = ref(false)  
// 商品信息本地数据源  
const localdata = ref({} as SkuPopupLocaldata)
```

```
const getGoodsByIdData = async () => {  
  // 获取商品详情  
  const res = await getGoodsByIdAPI(query.id)  
  // 准备SKU组件数据格式  
  localdata.value = {  
    _id: res.result.id,  
    goods_thumb: res.result.mainPictures[0],  
    name: res.result.name,  
    spec_list: res.result.specs.map((v) => ({  
      name: v.name, list: v.values  
    })),  
    sku_list: res.result.skus.map((v) => ({  
      _id: v.id,  
      goods_id: res.result.id,  
      goods_name: res.result.name,  
      image: v.picture,  
      price: v.price * 100, // 注意: 价格需要 * 100  
      stock: v.inventory,  
      sku_name_arr: v.specs.map((v) => v.valueName),  
    })),  
  }  
}
```



SKU模块 - 打开弹窗交互

打开SKU弹窗



设置按钮模式



微调组件样式

参数	说明	类型	默认值	可选值
mode	模式 1:都显示 2:只显示购物车 3:只显示立即购买	Number	1	1、2、3

```
<vk-data-goods-sku-popup  
  v-model="isShowSku"  
  :localdata="localdata"  
  :mode="mode"  
  add-cart-background-color="#FFA868"  
  buy-now-background-color="#27BA9B"  
>  
</>
```

```
// 弹窗模式  
enum SkuMode {  
  Both = 1,  
  Cart = 2,  
  Buy = 3,  
}  
const mode = ref<SkuMode>(SkuMode.Both)  
// 打开SKU弹窗  
const openSkuPopup = (val: SkuMode) => {  
  // 修改弹窗变量  
  isShowSku.value = true  
  // 修改按钮模式  
  mode.value = val  
}
```

```
<view @tap="openSkuPopup(SkuMode.Both)">  
  <text class="label">选择</text>  
  <text class="text">请选择商品规格</text>  
</view>  
  
<view @tap="openSkuPopup(SkuMode.Cart)">  
  加入购物车  
</view>  
  
<view @tap="openSkuPopup(SkuMode.Buy)">  
  立即购买  
</view>
```



SKU模块 - 被选中的值

获取SKU组件实例



计算被选中的值



渲染到页面



微调组件样式

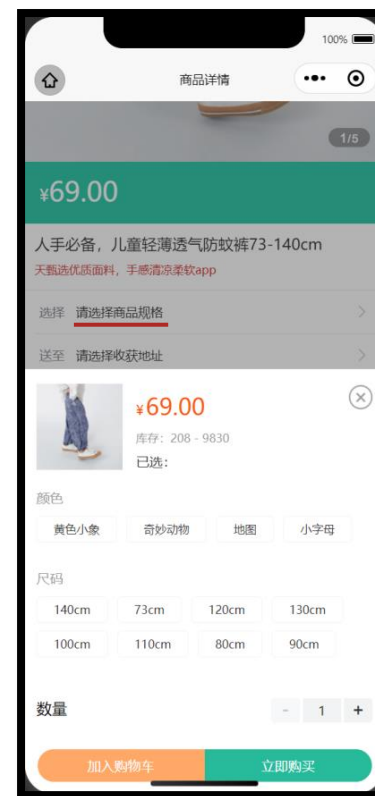
黄色小象

奇妙动物

```
<vk-data-goods-sku-popup
  v-model="isShowSku"
  :localdata="localdata"
  :mode="mode"
  add-cart-background-color="#FFA868"
  buy-now-background-color="#27BA9B"
  ref="skuPopupRef"
  :actived-style="{
    color: '#27BA9B',
    borderColor: '#27BA9B',
    backgroundColor: '#E9F8F5',
  }"
/>
```

```
// SKU组件实例
const skuPopupRef = ref<SkuPopupInstance>()
// 被选中的值
const selectArrText = computed(() => {
  return skuPopupRef.value?.selectArr?.join(' ').trim() || '请选择商品规格'
})
```

```
<view @tap="openSkuPopup(SkuMode.Both)" class="item arrow">
  <text class="label">选择</text>
  <text class="text">{{ selectArrText }} </text>
</view>
```



SKU模块 - 加入购物车

加入购物车事件



调用API接口



成功提示



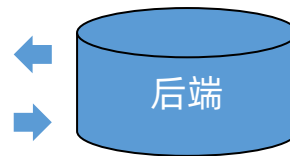
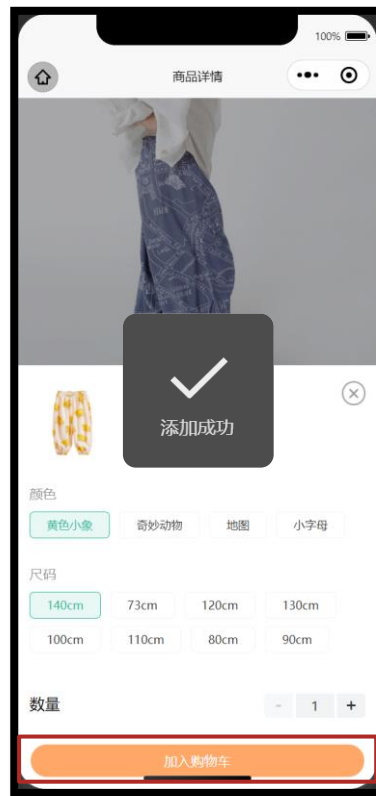
关闭SKU弹窗

```
<vk-data-goods-sku-popup  
  ...省略  
  @add-cart="onAddCart"  
>
```

```
/**  
 * 加入购物车  
 * @param data 请求参数  
 */  
export const postMemberCartAPI = (data: { skuId: string; count: number }) => {  
  return http({  
    method: 'POST',  
    url: '/member/cart',  
    data,  
  })  
}
```

```
// 加入购物车  
const onAddCart = async (ev: SkuPopupEvent) => {  
  // 调用接口  
  await postMemberCartAPI({ skuId: ev._id, count: ev.buy_num })  
  // 成功提示  
  uni.showToast({ title: '添加成功' })  
  // 关闭SKU弹窗  
  isShowSku.value = false  
}
```

事件名	说明	回调参数
add-cart	点击添加到购物车时（需选择完SKU才会触发）	event: 当前选择的sku数据
buy-now	点击立即购买时（需选择完SKU才会触发）	event: 当前选择的sku数据



购物车模块 - 列表渲染

页面静态结构



获取会员store



条件渲染



购物车列表API



初始化调用



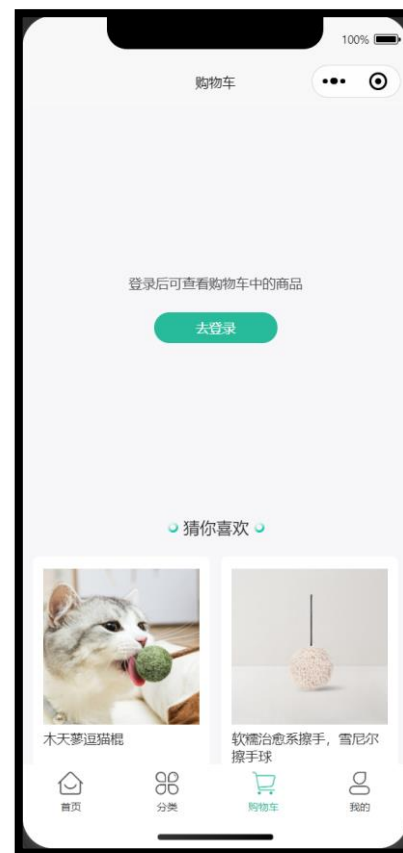
列表渲染

```
// 获取会员store
const memeberStore = useMemberStore()
// 获取购物车数据
const cartList = ref<CartItem[]>([])
const getMemberCartData = async () => {
  const res = await getMemberCartAPI()
  cartList.value = res.result
}
// 初始化调用
onShow(() => {
  if (memeberStore.profile) getMemberCartData()
})
```

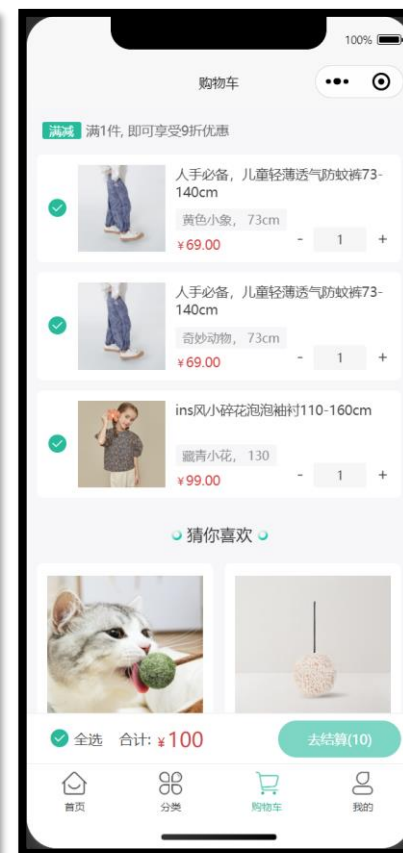
```
<!-- 已登录：显示购物车 -->
<template v-if="memeberStore.profile">
  <!-- 购物车列表 -->
  <view class="cart-list" v-if="cartList.length">
    <uni-swipe-action>
      <uni-swipe-action-item
        v-for="item in cartList"
        :key="item.skuId"
      >
        <view class="goods">
          ...商品信息
        </view>
      </uni-swipe-action-item>
    </uni-swipe-action>
  </view>
</template>
```

```
/**
 * 获取购物车列表
 */
export const getMemberCartAPI = () => {
  return http<CartItem[]>({
    method: 'GET',
    url: '/member/cart',
  })
}
```

```
/** 购物车类型 */
export type CartItem = {
  id: string
  skuId: string
  name: string
  picture: string
  count: number
  price: number
  nowPrice: number
  stock: number
  selected: boolean
  attrsText: string
  isEffective: boolean
}
```



未登录



已登录

温馨提示：微信小程序是多页面应用，登录成功后可通过 `uni.navigateBack()` 直接返回上一页。

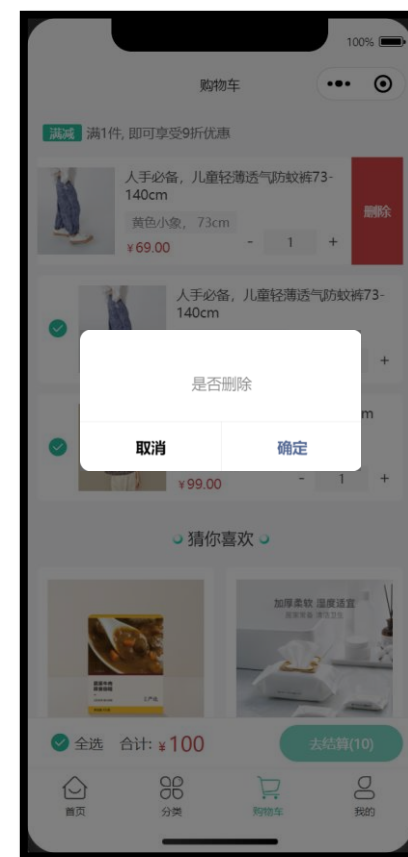
购物车模块 - 删除单品

购物车删除API ➡ 按钮绑定事件 ➡ 弹窗二次确认 ➡ 调用删除API ➡ 重新获取列表

```
/**
 * 删除/清空购物车单品
 * @param data 请求体参数, ids为skuId集合
 */
export const deleteMemberCartAPI = (data: { ids: string[] }) => {
  return http({
    method: 'DELETE',
    url: '/member/cart',
    data,
  })
}
```

```
<!-- 滑动操作分区 -->
<uni-swipe-action>
  <!-- 滑动操作项 -->
  <uni-swipe-action-item
    v-for="item in cartList"
    :key="item.skuId"
  >
    <!-- 右侧删除按钮 -->
    <template #right>
      <view class="cart-swipe-right">
        <button @tap="onDeleteCart(item.skuId)">删除</button>
      </view>
    </template>
  </uni-swipe-action-item>
</uni-swipe-action>
```

```
// 点击删除按钮
const onDeleteCart = (skuId: string) => {
  // 显示模态弹窗
  uni.showModal({
    content: '是否删除',
    success: async (res) => {
      if (res.confirm) {
        // 删除购物车商品
        await deleteMemberCartAPI({ ids: [skuId] })
        // 获取购物车数据
        getMemberCartData()
      }
    },
  })
}
```



购物车模块 - 修改单品数量

步进器组件



类型声明文件



属性绑定



事件绑定



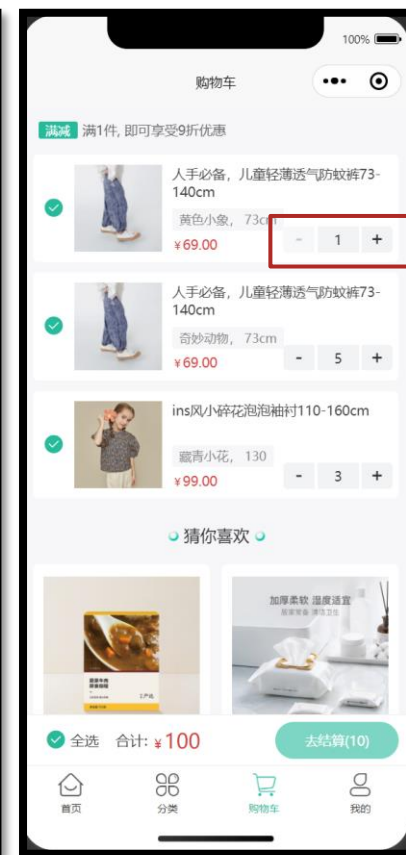
调用单品修改API

```
<!-- 商品数量 -->
<vk-data-input-number-box v-model="item.count" :min="1" :max="item.stock" :index="item.skuId" @change="onChangeCount" />
```

```
/**
 * 修改购物车单品
 * @param skuId 路径参数
 * @param data
 */
export const putmemberCartBySkuIdAPI = (
  skuId: string,
  data: { selected?: boolean; count?: number }
) => {
  return http({
    method: 'PUT',
    url: `/member/cart/${skuId}`,
    data,
  })
}
```

```
// 修改步进器
const onChangeCount = (ev: InputNumberBoxEvent) => {
  // 后端数据更新
  putmemberCartBySkuIdAPI(ev.index, { count: ev.value })
}
```

```
/** 步进器 */
export type InputNumberBox = Component<InputNumberBoxProps>
/** 步进器实例 */
export type InputNumberBoxInstance = InstanceType<InputNumberBox>
/** 步进器属性 */
export type InputNumberBoxProps = {
  modelValue: number
  min: number
  max: number
  index: string
  onChange: (event: InputNumberBoxEvent) => void
}
/** 步进器事件对象 */
export type InputNumberBoxEvent = {
  value: number
  index: string
}
/** 全局组件类型声明 */
declare module '@vue/runtime-core' {
  export interface GlobalComponents {
    'vk-data-input-number-box': InputNumberBox
  }
}
```



购物车模块 - 修改选中状态

点击单品选中



调用修改单品选中API



计算全选状态



点击全选选中



调用修改全选状态API

```
<!-- 单品选中状态 -->
<text class="checkbox" :class="{ checked: item.selected }" @tap="onChangeSelected(item)" ></text>
<!-- 底部全选状态 -->
<text class="all" :class="{ checked: isSelectedAll }" @tap="onChangeSelectedAll">全选</text>
```

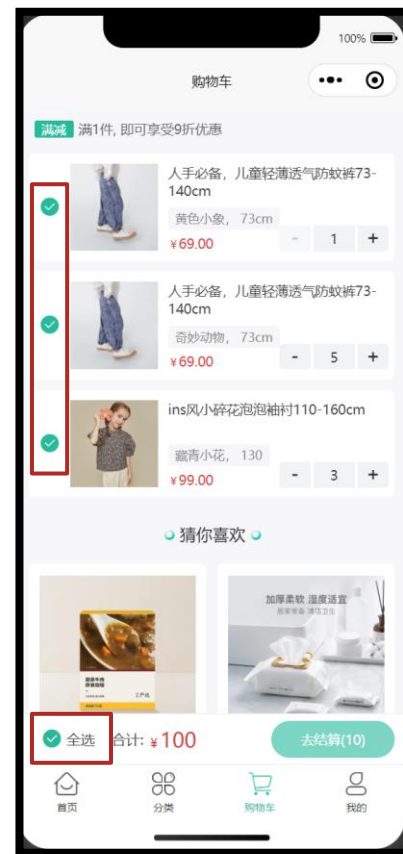
```
// 修改选中状态
const onChangeSelected = (item: CartItem) => {
  // 取反选中状态
  item.selected = !item.selected
  // 后端数据更新
  putmemberCartBySkuIdAPI(item.skuId, { selected: item.selected })
}

// 计算全选状态
const isSelectedAll = computed(() => {
  return cartList.value.length && cartList.value.every((v) => v.selected)
})

// 修改全选状态
const onChangeSelectedAll = () => {
  // 取反计算结果
  const _isSelectedAll = !isSelectedAll.value
  // 前端数据更新
  cartList.value.forEach((item) => {
    item.selected = _isSelectedAll
  })
  // 后端数据更新
  putmemberCartSelectedAPI({ selected: _isSelectedAll })
}
```

```
/**修改购物车商品 - 单品修改 */
export const putmemberCartBySkuIdAPI = (
  skuId: string,
  data: { selected?: boolean; count?:
  number },
) => {
  return http({
    method: 'PUT',
    url: `/member/cart/${skuId}`,
    data,
  })
}
```

```
/** 购物车全选/取消全选 */
export const putmemberCartSelectedAPI = (
  data: { selected: boolean }
) => {
  return http({
    method: 'PUT',
    url: `/member/cart/selected`,
    data,
  })
}
```



购物车模块 - 底部结算信息

计算选中单品列表



计算选中总件数



计算选中总金额

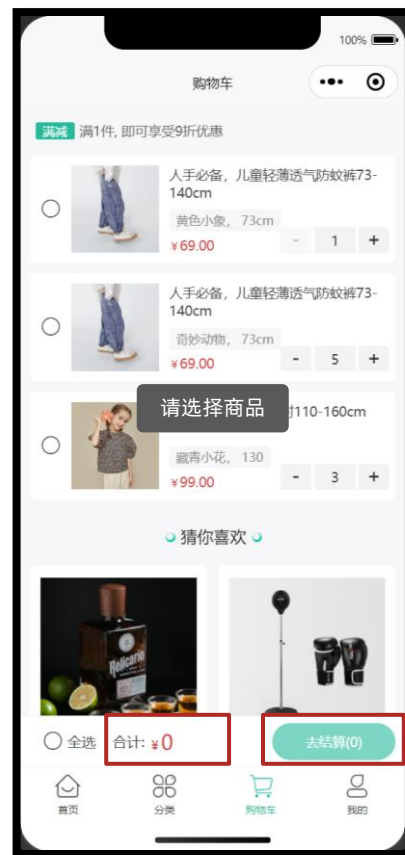


结算按钮交互

```
// 计算已选中单品列表
const selectedCartList = computed(() => {
  return cartList.value.filter((v) => v.selected)
})
// 计算选中总件数
const selectedCartListCount = computed(() => {
  return selectedCartList.value.reduce((sum, v) => sum + v.count, 0)
})
// 计算选中总金额
const selectedCartListMoney = computed(() => {
  return selectedCartList.value.reduce((sum, v) => sum + v.count * v.nowPrice, 0).toFixed(2)
})
```

```
<!-- 吸底工具栏 -->
<view class="toolbar">
  <text class="text">合计:</text>
  <text class="amount">{{ selectedCartListMoney }}</text>
  <view class="button-group">
    <view
      class="button payment-button"
      :class="{ disabled: selectedCartListCount === 0 }"
      @tap="onPayment"
    >
      去结算({{ selectedCartListCount }})
    </view>
  </view>
</view>
</view>
```

```
// 点击去结算按钮
const onPayment = () => {
  if (selectedCartListCount.value === 0) {
    uni.showToast({
      icon: 'none',
      title: '请选择商品'
    })
    return
  }
  uni.showToast({ title: '等待完善' })
}
```



购物车模块 - 两个购物车页面

tabBar页: 小程序跳转到 tabBar 页面时，会关闭其他所有非 tabBar 页面，所以小程序的 **tabBar 页没有后退按钮**。



```
// 两个购物车页
<script setup lang="ts">
import CartMain from './components/CartMain.vue'
</script>

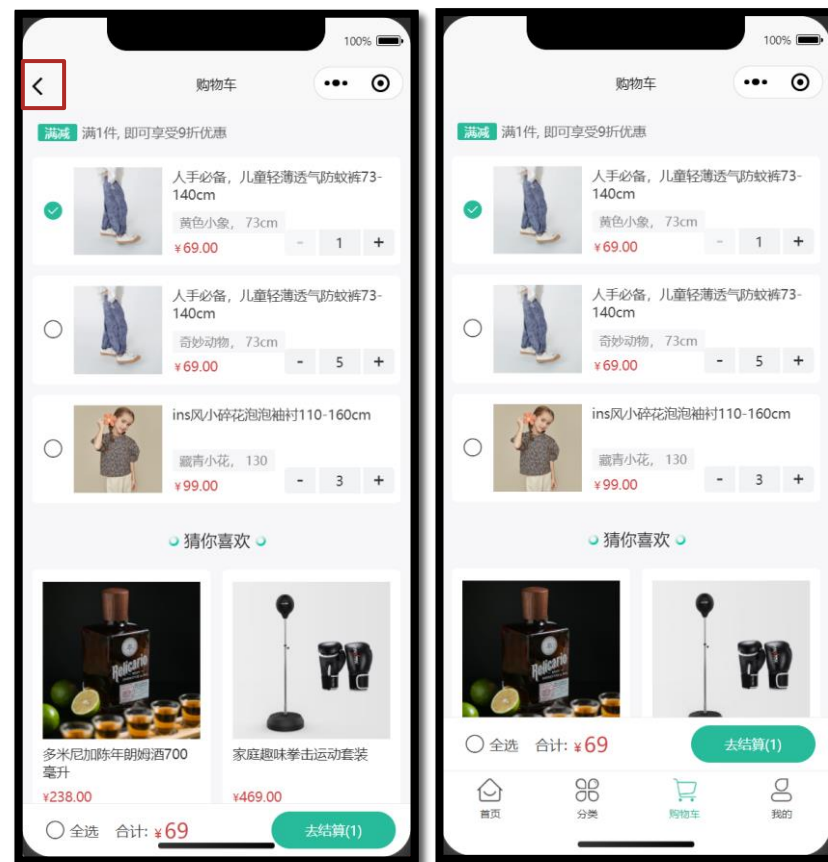
<template>
  <CartMain />
</template>
```

```
// 商品详情页
<navigator url="/pages/cart/cart2">
  <text class="icon-cart"></text>购物车
</navigator>
```

```
// pages.json
{
  "path": "pages/cart/cart",
  "style": {
    "navigationBarTitleText": "购物车"
  },
  + {
  +   "path": "pages/cart/cart2",
  +   "style": {
  +     "navigationBarTitleText": "购物车"
  +   }
  + },
}
```

作业:

1. 实现猜你喜欢分页加载
2. 底部工具栏安全区适配



普通页

tabBar页



传智教育旗下高端IT教育品牌