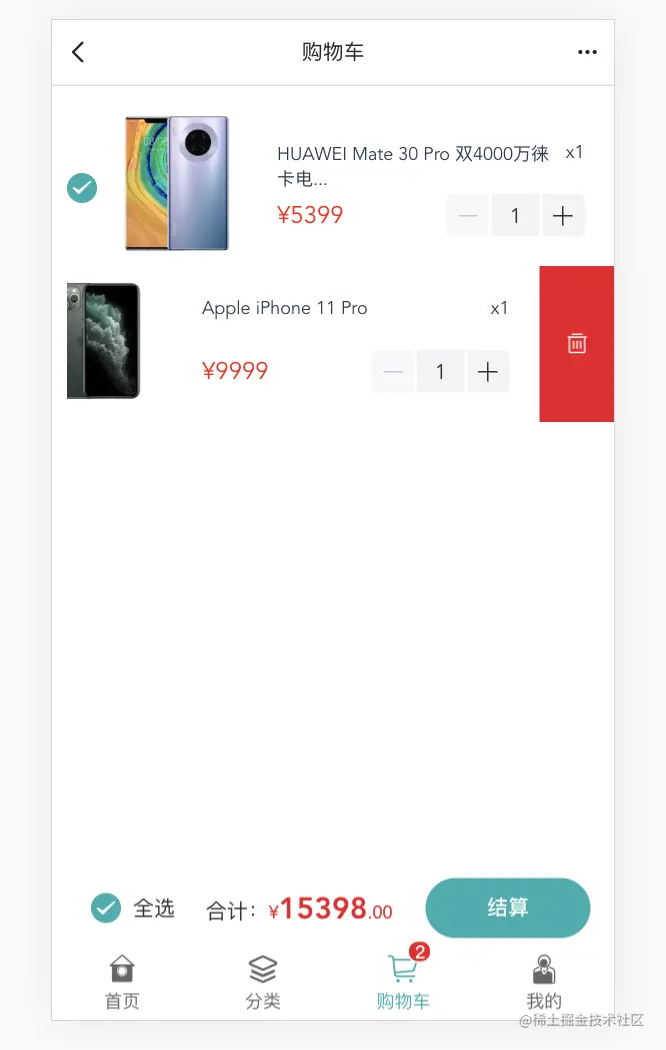
## 前言

本章节直接进入购物车模块环节，将会用到的一些 ES6 方法，以及 Vue 的 computed 计算属性，以及我们上一章节创建的购物车数量全局状态的变化。我们根据现成的项目分析购物车的布局和逻辑。

## 购物车需求分析

在商品详情页中我们可以点击底部按钮 「加入购物车」，将相应的商品通过后端接口添加至购物列表中。当我们点击底部「购物车」Tab，就会进入购物车页面，如下图所示：



该页面会设计如下功能：

* 基础页面展示
* 购物项列表展示
* 购物项数量修改
* 购物项删除
* 购物项选择操作
* 结算功能

相关交互事项记录如下：

页面中需要展示的购物车内的商品列表通过 getCart 接口获取，「购物车」页面左边栏是勾选要进行结算的商品，这里需要涉及到价格的变化，通过计算属性 computed 监听购物车商品列表的被勾选情况，动态的变化合计金额。

勾选「全选」，选中所有购物车商品并计算出总价，勾掉「全选」商品，则取消选择且合计金额清零。

单项商品左滑，调出 Vant 组件为我们写好的删除选项，组件库的作用得到放大，若是自己手写这段代码当然可以，只是费些时间罢了，效果可能还没有组件库封装的好。

点击「结算」按钮，我们将会调用前往订单生成页面，也就是需要选择地址等信息，支付这笔订单，这部分内容我们下一章节再分析。

## 购物车列表

首先介绍本章节需要用到的 Vant 组件：

* SwipeCell
* CheckboxGroup
* Checkbox
* SubmitBar
* Stepper

我们先将它们进行全局注册，代码如下：

**import** **Vue** **from** 'vue'**import** md5 **from** 'js-md5';**import** **App** **from** './App.vue'**import** router **from** './router'**import** store **from** './store'**import** {**Button**, **Form**, **Field**, **Toast**, **Icon**, **Swipe**, **SwipeItem**, **SwipeCell**, **PullRefresh**, **List**, **Tab**, **Tabs**, **SwipeCell**, **CheckboxGroup**, **Checkbox**, **SubmitBar**, **Stepper** } **from** 'vant'**import** 'lib-flexible/flexible'

**Vue**.**use**(**Form**).**use**(**Field**).**use**(**Toast**).**use**(**Button**).**use**(**Icon**).**use**(**Swipe**).**use**(**SwipeItem**).**use**(**SwipeCell**).**use**(**PullRefresh**).**use**(**List**).**use**(**Tab**).**use**(**Tabs**).**use**(**SwipeCell**).**use**(**CheckboxGroup**).**use**(**Checkbox**).**use**(**SubmitBar**).**use**(**Stepper**)

**Vue**.config.productionTip = false

**new** **Vue**({

router,

store,

render: h => **h**(**App**)

}).$mount('#app')

**Vue**.**prototype**.$md5 = md5;window.vRouter = router

注册完之后，我们进入 views/Cart.vue 页面，进行购物车页面的布局排版，首先引入头部信息：

<template>

<div class="cart-box">

<s-header :title="'购物车'"></s-header>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

export default {

name: 'Cart',

components: {

sHeader

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.cart-box {

}

</style>

再对购物车的商品列表及商品项进行编写，主要展示内容为商品的基本信息及想要购买的数量，代码如下：

<template>

<div class="cart-box">

<s-header :title="'购物车'"></s-header>

<div class="cart-body">

<van-swipe-cell :right-width="50" v-for="(item, index) in list" :key="index">

<div class="good-item">

<van-checkbox :name="item.cartItemId" />

<div class="good-img"><img :src="`//api.newbee.ltd${item.goodsCoverImg}`" alt=""></div>

<div class="good-desc">

<div class="good-title">

<span>{{ item.goodsName }}</span>

<span>x{{ item.goodsCount }}</span>

</div>

<div class="good-btn">

<div class="price">¥{{ item.sellingPrice }}</div>

<van-stepper

integer

:min="1"

:value="item.goodsCount"

:name="item.cartItemId"

async-change

@change="onChange"

/>

</div>

</div>

</div>

<van-button

slot="right"

square

icon="delete"

type="danger"

class="delete-button"

@click="deleteGood(item.cartItemId)"

/>

</van-swipe-cell>

</div>

</div>

</template>

<script>

import { Toast } from 'vant'

import sHeader from '@/components/SimpleHeader'

import { getCart } from '../service/cart'

export default {

name: 'Cart',

components: {

sHeader

},

data() {

return {

list: [], // 购物车商品列表

result: [], // 选中的购物车商品 id 数组

}

},

mounted() {

this.init()

},

methods: {

async init() {

// 加载中禁止点击

Toast.loading({ message: '加载中...', forbidClick: true });

// 获取购物车商品数据

const { data } = await getCart({ pageNumber: 1 })

this.list = data

this.result = data.map(item => item.cartItemId)

Toast.clear()

},

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.cart-box {

.cart-body {

margin: 60px 0 100px 0;

padding-left: 10px;

.good-item {

display: flex;

.good-img {

img {

.wh(100px, 100px)

}

}

.good-desc {

display: flex;

flex-direction: column;

justify-content: space-between;

flex: 1;

padding: 20px;

.good-title {

display: flex;

justify-content: space-between;

}

.good-btn {

display: flex;

justify-content: space-between;

.price {

font-size: 16px;

color: red;

line-height: 28px;

}

.van-icon-delete {

font-size: 20px;

margin-top: 4px;

}

}

}

}

.delete-button {

width: 50px;

height: 100%;

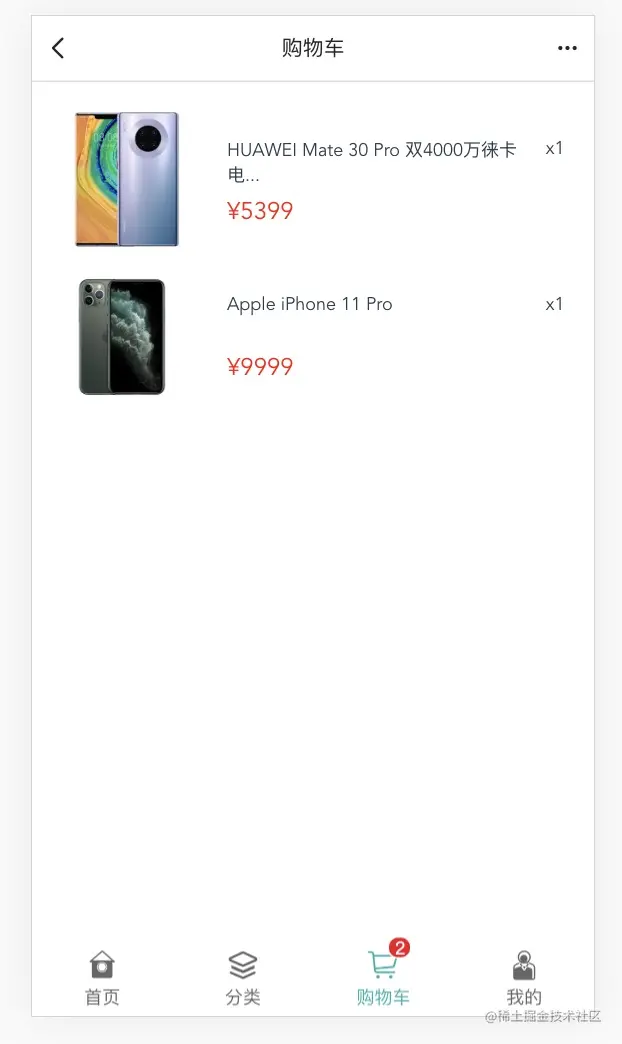
}

}

}

</style>

浏览器效果如下所示：



此时购物车的商品单项已经加上，需要在购物车的左边栏加上多选按钮，Vant 组件中已经为我们提供好了现成的组件 — CheckboxGroup 。

官网使用示例如下：

<van-checkbox-group v-model="result" ref="checkboxGroup">

<van-checkbox name="a">复选框 a</van-checkbox>

<van-checkbox name="b">复选框 b</van-checkbox>

<van-checkbox name="c">复选框 c</van-checkbox>

</van-checkbox-group>

<van-button type="primary" @click="checkAll">全选</van-button>

<van-button type="info" @click="toggleAll">反选</van-button>

export default {

data() {

return {

result: [],

};

},

methods: {

checkAll() {

this.$refs.checkboxGroup.toggleAll(true);

},

toggleAll() {

this.$refs.checkboxGroup.toggleAll();

},

},

};

van-checkbox-group 组件包裹 van-checkbox 组件，形成一个「组」的概念。

包裹下的 van-checkbox 可以通过 result 值控制，所以我们前面的代码创建了一个 result 变量用于存储勾选的购物车商品，在结算的时候我们需要把勾选的购物车内的商品删除，所以我们不能在购物车页面内点击「结算」按钮的时候进行删除勾选的商品，因为在后续生成订单页面中，用户可能会选择回退或者返回「首页」。

具体代码如下：

<template>

<div class="cart-box">

<s-header :title="'购物车'"></s-header>

<div class="cart-body">

<van-checkbox-group @change="groupChange" v-model="result" ref="checkboxGroup">

<van-swipe-cell :right-width="50" v-for="(item, index) in list" :key="index">

<div class="good-item">

<van-checkbox :name="item.cartItemId" />

<div class="good-img"><img :src="`//api.newbee.ltd${item.goodsCoverImg}`" alt=""></div>

<div class="good-desc">

<div class="good-title">

<span>{{ item.goodsName }}</span>

<span>x{{ item.goodsCount }}</span>

</div>

<div class="good-btn">

<div class="price">¥{{ item.sellingPrice }}</div>

<van-stepper

integer

:min="1"

:value="item.goodsCount"

:name="item.cartItemId"

async-change

@change="onChange"

/>

</div>

</div>

</div>

<van-button

slot="right"

square

icon="delete"

type="danger"

class="delete-button"

@click="deleteGood(item.cartItemId)"

/>

</van-swipe-cell>

</van-checkbox-group>

</div>

</div>

</template>

<script>

import { Toast } from 'vant'

import sHeader from '@/components/SimpleHeader'

import { getCart, deleteCartItem, modifyCart } from '../service/cart'

export default {

name: 'Cart',

components: {

sHeader

},

data() {

return {

list: [],

result: [],

}

},

mounted() {

this.init()

},

methods: {

async init() {

Toast.loading({ message: '加载中...', forbidClick: true });

const { data } = await getCart({ pageNumber: 1 })

this.list = data

this.result = data.map(item => item.cartItemId)

Toast.clear()

},

async deleteGood(id) {

const { data } = await deleteCartItem(id)

this.$store.dispatch('updateCart')

this.init()

},

// 单项购买数量变化回调

async onChange(value, detail) {

if (this.list.filter(item => item.cartItemId == detail.name)[0].goodsCount == value) return

Toast.loading({ message: '修改中...', forbidClick: true });

const params = {

cartItemId: detail.name,

goodsCount: value

}

const { data } = await modifyCart(params)

this.list.forEach(item => {

if (item.cartItemId == detail.name) {

item.goodsCount = value

}

})

Toast.clear();

},

// 多选变化是，整组的回调

groupChange(result) {

if (result.length == this.list.length) {

this.checkAll = true

} else {

this.checkAll = false

}

this.result = result

},

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.cart-box {

.cart-body {

margin: 60px 0 100px 0;

padding-left: 10px;

.good-item {

display: flex;

.good-img {

img {

.wh(100px, 100px)

}

}

.good-desc {

display: flex;

flex-direction: column;

justify-content: space-between;

flex: 1;

padding: 20px;

.good-title {

display: flex;

justify-content: space-between;

}

.good-btn {

display: flex;

justify-content: space-between;

.price {

font-size: 16px;

color: red;

line-height: 28px;

}

.van-icon-delete {

font-size: 20px;

margin-top: 4px;

}

}

}

}

.delete-button {

width: 50px;

height: 100%;

}

}

}

</style>

对应购物车的接口脚本如下：

**import** axios **from** '../utils/axios'

**export** **function** **addCart**(params) {

**return** axios.**post**('/shop-cart', params);

}

**export** **function** **getCart**(params) {

**return** axios.**get**('/shop-cart', { params });

}

**export** **function** **deleteCartItem**(id) {

**return** axios.**delete**(`/shop-cart/${id}`);

}

**export** **function** **modifyCart**(params) {

**return** axios.**put**('/shop-cart', params);

}

添加了修改购物项数量、删除购物项等接口。

浏览效果图时，我发现多选框的主题色没有变，还是默认的 Vant 组件颜色，我们来点“黑魔法”，将主题色变成我们商城的颜色，在底部再加一个 style 标签，但是不要加 scoped 属性，如下所示：

<style lang="less">

@import '../common/style/mixin';

.van-checkbox\_\_icon--checked .van-icon {

background-color: @primary;

border-color: @primary;

}

</style>

效果图如下所示：



## 购物车底部加个变化

购物车列表基本上已经收尾，我们还需添加购物车底部价格状态变化，以及结算按钮。

在上述代码中我们有一个变量 result 用于记录已选中的商品 id，我们利用它结合 computed 属性，来动态计算价格变量。

首先我们先声明一个变量 total 用于计算总价的变化，再通过 result 列表，过滤出对应商品的价格进行累加，顺带把全选状态也加上，代码及相关注释如下：

<template>

<div class="cart-box">

<s-header :title="'购物车'"></s-header>

<div class="cart-body">

<van-checkbox-group @change="groupChange" v-model="result" ref="checkboxGroup">

<van-swipe-cell :right-width="50" v-for="(item, index) in list" :key="index">

<div class="good-item">

<van-checkbox :name="item.cartItemId" />

<div class="good-img"><img :src="`//api.newbee.ltd${item.goodsCoverImg}`" alt=""></div>

<div class="good-desc">

<div class="good-title">

<span>{{ item.goodsName }}</span>

<span>x{{ item.goodsCount }}</span>

</div>

<div class="good-btn">

<div class="price">¥{{ item.sellingPrice }}</div>

<van-stepper

integer

:min="1"

:value="item.goodsCount"

:name="item.cartItemId"

async-change

@change="onChange"

/>

</div>

</div>

</div>

<van-button

slot="right"

square

icon="delete"

type="danger"

class="delete-button"

@click="deleteGood(item.cartItemId)"

/>

</van-swipe-cell>

</van-checkbox-group>

</div>

<van-submit-bar

v-if="list.length > 0"

class="submit-all"

:price="total \* 100"

button-text="结算"

@submit="onSubmit"

>

<van-checkbox @click="allCheck" v-model="checkAll">全选</van-checkbox>

</van-submit-bar>

</div>

</template>

<script>

import { Toast } from 'vant'

import sHeader from '@/components/SimpleHeader'

import { getCart, deleteCartItem } from '../service/cart'

export default {

name: 'Cart',

components: {

sHeader

},

data() {

return {

list: [],

result: [],

checkAll: true

}

},

mounted() {

this.init()

},

computed: {

total: function() {

let sum = 0

let \_list = this.list.filter(item => this.result.includes(item.cartItemId))

\_list.forEach(item => {

sum += item.goodsCount \* item.sellingPrice

})

return sum

}

},

methods: {

async init() {

Toast.loading({ message: '加载中...', forbidClick: true });

const { data } = await getCart({ pageNumber: 1 })

this.list = data

this.result = data.map(item => item.cartItemId)

Toast.clear()

},

async deleteGood(id) {

const { data } = await deleteCartItem(id)

this.$store.dispatch('updateCart')

this.init()

},

async onChange(value, detail) {

if (this.list.filter(item => item.cartItemId == detail.name)[0].goodsCount == value) return

Toast.loading({ message: '修改中...', forbidClick: true });

const params = {

cartItemId: detail.name,

goodsCount: value

}

const { data } = await modifyCart(params)

this.list.forEach(item => {

if (item.cartItemId == detail.name) {

item.goodsCount = value

}

})

Toast.clear();

},

groupChange(result) {

if (result.length == this.list.length) {

this.checkAll = true

} else {

this.checkAll = false

}

this.result = result

},

async onSubmit() {

const params = JSON.stringify(this.result)

this.$router.push({ path: `create-order?cartItemIds=${params}` })

},

allCheck(value) {

// 判断 checkAll ，如果已是全选状态，checkAll 将变为false，所以清空 result 内的变量，价格变为 0

// 如果是非全选状态checkAll 将变为 true，直接将 list 下的 id 塞进 result 变量里，total 会自动变为相应的价格

if (this.checkAll) {

this.result = this.list.map(item => item.cartItemId)

} else {

this.result = []

}

}

}

}

</script>

<style lang="less" scoped>

@import '../common/style/mixin';

.cart-box {

.cart-body {

margin: 60px 0 100px 0;

padding-left: 10px;

.good-item {

display: flex;

.good-img {

img {

.wh(100px, 100px)

}

}

.good-desc {

display: flex;

flex-direction: column;

justify-content: space-between;

flex: 1;

padding: 20px;

.good-title {

display: flex;

justify-content: space-between;

}

.good-btn {

display: flex;

justify-content: space-between;

.price {

font-size: 16px;

color: red;

line-height: 28px;

}

.van-icon-delete {

font-size: 20px;

margin-top: 4px;

}

}

}

}

.delete-button {

width: 50px;

height: 100%;

}

}

.submit-all {

margin-bottom: 50px;

.van-checkbox {

margin-left: 10px

}

.van-submit-bar\_\_text {

margin-right: 10px

}

.van-submit-bar\_\_button {

background: @primary;

}

}

}

</style>

<style lang="less">

@import '../common/style/mixin';

.van-checkbox\_\_icon--checked .van-icon {

background-color: @primary;

border-color: @primary;

}

</style>

最后我们加一个空状态页面，当购物车为空时显示，同时放置一个跳转按钮，点击跳转至商城首页：

<div class="empty" v-if="!list.length">

<van-icon name="smile-o" />

<div class="title">购物车空空空如也</div>

<van-button color="#1baeae" type="primary" @click="goTo" block>前往首页</van-button>

</div>

样式代码：

.empty {

width: 50%;

margin: 0 auto;

text-align: center;

margin-top: 200px;

.van-icon-smile-o {

font-size: 50px;

}

.title {

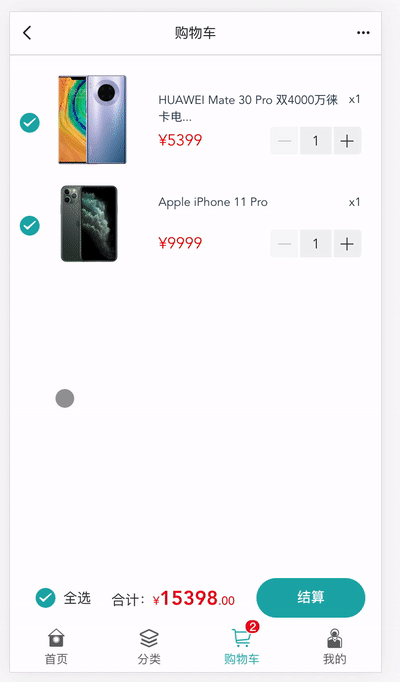
font-size: 16px;

margin-bottom: 20px;

}

}

最后咱们看看浏览器展示效果：



购物项为空时显示缺省页面，购物项不为空时则在当前的购物车页面显示购物项列表。