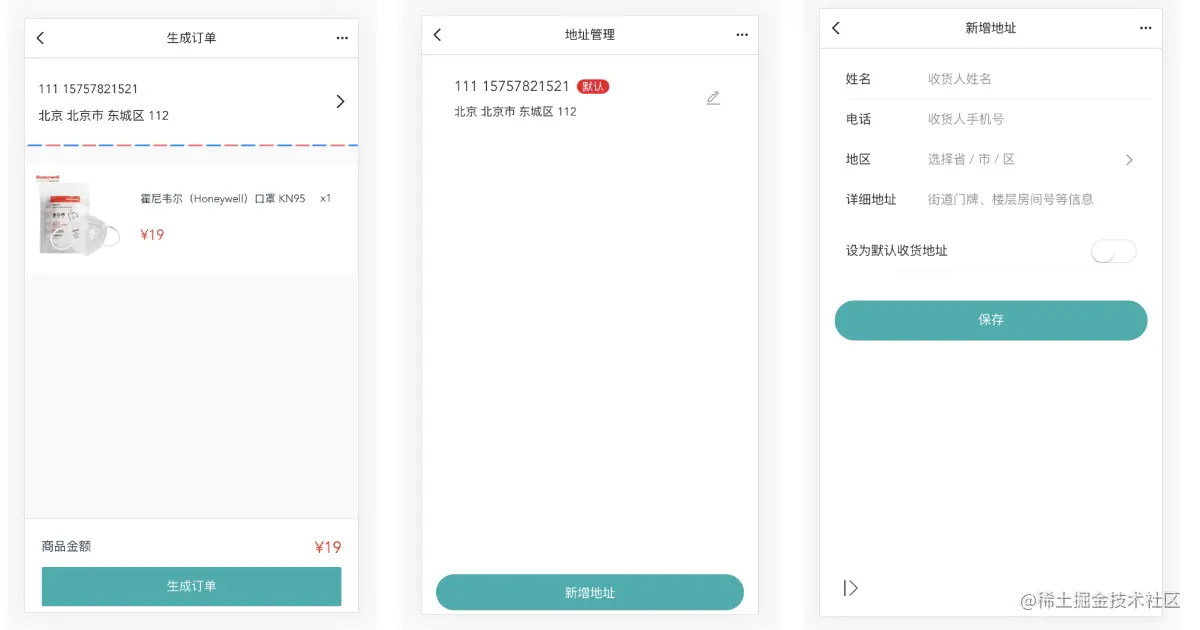
## 前言

在购物车页面中选择自己需要的商品之后，点击结算按钮即可跳转至生产订单页面，在这个页面进行“提交订单”的操作，不过生成订单数据之前，用户还需要填写相关信息，如地址及个人信息收货信息等，所以本章节将地址管理页面和确认订单页面放在一起，这样逻辑上也不会断层。

地址栏的管理我们会用到一个全国省市区的静态列表，这块内容是相对固定的，直接在前端处理即可，无需联调后端接口。在支付这块流程由于没有公司资质，我们就是模拟一下支付流程，用户在点击支付按钮后即表示支付成功。

## 需求分析



生成订单流程如上图所示，这里牵涉到确认订单页面和个人地址管理页面，交互如下：

点击图 1 头部地址信息区域，进入图 2 地址栏列表页面进行地址选择，若是没有自己需要的地址，可点击新增地址跳转到图 3 进行地址的添加。也可点击图 2 的铅笔图标进行当前地址的修改，同样是跳到图 3，只不过会在浏览器地址栏携带当前地址 id 获取地址详情。选择好需要的地址，带上地址参数再跳回图 1 确认订单页面，点击生成订单后，将会调用生成订单接口，创建一个新的订单，这时我们再跳转到我的订单列表页面，点击生成订单即可进行“提交订单”的操作。

## 生成订单页

首先我们 src/views 文件夹下新建文件 CreateOrder.vue 。

先来介绍一下本章节需要用到的组件，做好全局注册工作，引入 Popup 组件，代码如下：

*// main.js***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**, **GoodsAction**, **GoodsActionIcon**, **GoodsActionButton**, **CheckboxGroup**, **Checkbox**, **SubmitBar**, **Stepper**, **Popup** } **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**(**GoodsAction**).**use**(**GoodsActionButton**)

.**use**(**GoodsActionIcon**).**use**(**CheckboxGroup**).**use**(**Checkbox**)

.**use**(**SubmitBar**).**use**(**Stepper**).**use**(**Popup**)

**Vue**.config.productionTip = false

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

router,

store,

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

}).$mount('#app')

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

然后我们开始布局，页面代码如下：

<template>

<div class="create-order">

<s-header :title="'生成订单'"></s-header>

<div class="address-wrap">

<div class="name">

<span>我是昵称</span>

<span>15758394732</span>

</div>

<div class="address">

浙江省 杭州市 西湖区 西湖底23号

</div>

<van-icon class="arrow" name="arrow" />

</div>

<div class="good">

<div class="good-item">

<div class="good-img"><img src='https://iph.href.lu/50x50?text=%E5%8D%A0%E4%BD%8D&fg=666666&bg=cccccc' alt=""></div>

<div class="good-desc">

<div class="good-title">

<span>小米10</span>

<span>x2</span>

</div>

<div class="good-btn">

<div class="price">¥2000</div>

</div>

</div>

</div>

</div>

<div class="pay-wrap">

<div class="price">

<span>商品金额</span>

<span>¥6000</span>

</div>

<van-button class="pay-btn" color="#1baeae" type="primary" block>生成订单</van-button>

</div>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

export default {

components: {

sHeader

},

}

</script>

<style lang="less" scoped>

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

.create-order {

background: #f9f9f9;

.address-wrap {

margin-bottom: 20px;

background: #fff;

position: relative;

margin-top: 44px;

font-size: 14px;

padding: 15px;

color: #222333;

.name, .address {

margin: 10px 0;

}

.arrow {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

font-size: 20px;

}

&::before {

position: absolute;

right: 0;

bottom: 0;

left: 0;

height: 2px;

background: -webkit-repeating-linear-gradient(135deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background: repeating-linear-gradient(-45deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background-size: 80px;

content: '';

}

}

.good {

margin-bottom: 120px;

}

.good-item {

padding: 10px;

background: #fff;

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;

}

}

}

}

.pay-wrap {

position: fixed;

bottom: 0;

left: 0;

width: 100%;

background: #fff;

padding: 10px 0;

padding-bottom: 50px;

border-top: 1px solid #e9e9e9;

>div {

display: flex;

justify-content: space-between;

padding: 0 5%;

margin: 10px 0;

font-size: 14px;

span:nth-child(2) {

color: red;

font-size: 18px;

}

}

.pay-btn {

position: fixed;

bottom: 7px;

right: 0;

left: 0;

width: 90%;

margin: 0 auto;

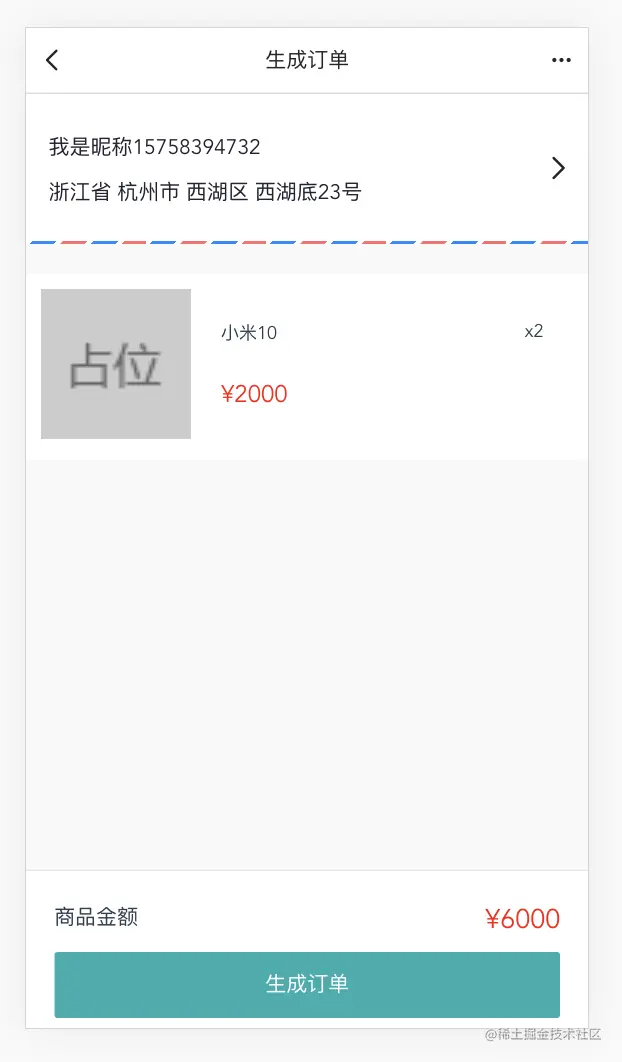
}

}

}

</style>

效果如下所示：



主要包括收货地址显示区域及购物项列表显示区域，页面最下方是订单的预估总价和“生成订单”功能按钮。

接下来接入订单详情数据，在这之前我们在上购物车章节中，点击「结算」按钮，跳转的地址就是上述页面地址，并且我们在地址栏携带了查询字符串，地址内容如下所示：

**async** **onSubmit**() {

**const** params = **JSON**.**stringify**(this.result)

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

},

我们将选中的购物车 id 数组通过 JSON.stringify 转化为字符串的形式传入 createOrder.vue 页面，我们通过 id 获取相应的商品详情，代码如下：

...

<div class="good">

<div class="good-item" v-for="(item, index) in cartList" :key="index">

<div class="good-img"><img :src="`//lmall.xinfeng.site${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>

</div>

</div>

</div>

</div>

...

...

<script>

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

import { setLocal, getLocal } from '@/common/js/utils'

import { Toast } from 'vant'

export default {

...

data() {

return {

cartList: [],

}

}

...

mounted() {

this.init()

},

methods: {

async init() {

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

// 获取查询参数内的 id

const { cartItemIds } = this.$route.query

// id 会本地存储，如果查询字符串 id 优先获取，若没有则获取本地存储的 ids

const \_cartItemIds = cartItemIds ? JSON.parse(cartItemIds) : JSON.parse(getLocal('cartItemIds'))

setLocal('cartItemIds', JSON.stringify(\_cartItemIds))

const { data: list } = await getByCartItemIds({ cartItemIds: \_cartItemIds.join(',') })

this.cartList = list

Toast.clear()

},

}

}

</script>

我们再次利用 computed 属性，计算商品的总价，如下所示：

computed: {

total: **function**() {

**let** sum = 0

this.cartList.**forEach**(item => {

sum += item.goodsCount \* item.sellingPrice

})

**return** sum

}

}

total 属性填入模板总价的位置，最后别忘了把接口加到 cart.js ：

**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);

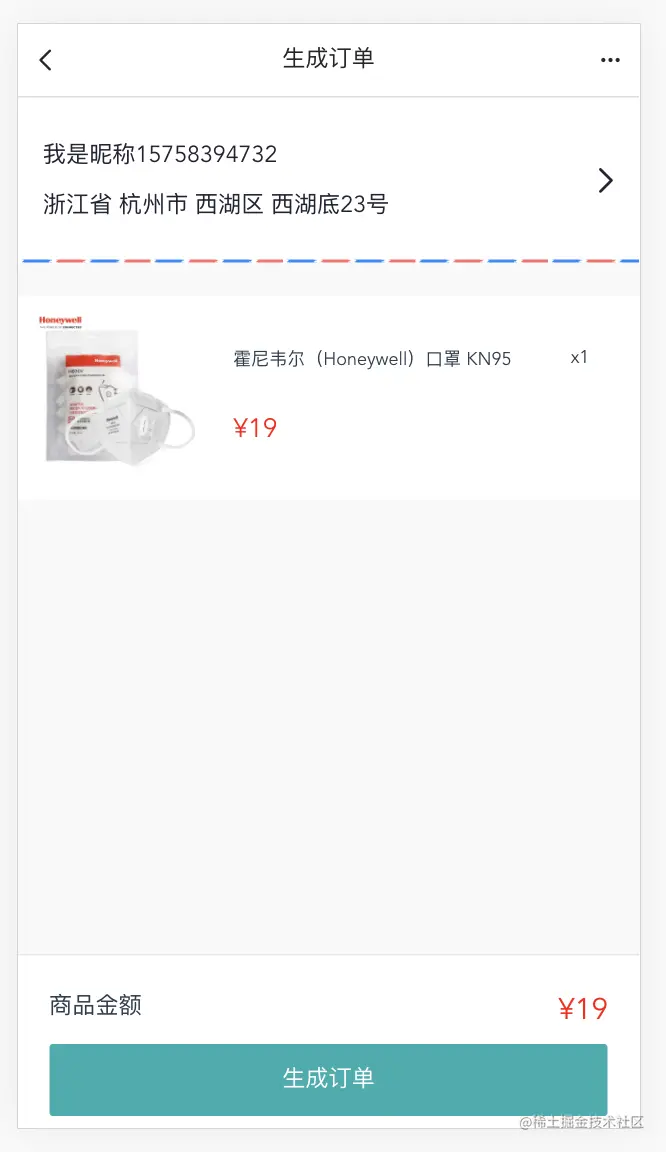
}

**export** **function** **getByCartItemIds**(params) {

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

}

打开浏览器刷新显示如下：



## 地址栏

头部的地址栏，有两种情况。

第一种是进入确认订单页面时，直接获取默认的地址接口，前提条件是地址列表设置了默认地址，否则直接返回一个空数据。

第二种则是从地址栏页面选择一个地址，跳回确认订单页面。此时链接会带上当前所选地址的 id，在确认订单页面获取到此 id，并根据 id 获取对应的地址详情，渲染页面。

分析完之后，我们直接将地址栏用到的接口，全部添加至 service/address.js 下：

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

**export** **function** **addAddress**(params) {

**return** axios.**post**('/address', params);

}

**export** **function** **EditAddress**(params) {

**return** axios.**put**('/address', params);

}

**export** **function** **DeleteAddress**(id) {

**return** axios.**delete**(`/address/${id}`);

}

**export** **function** **getDefaultAddress**() {

**return** axios.**get**('/address/default');

}

**export** **function** **getAddressList**() {

**return** axios.**get**('/address', { pageNumber: 1, pageSize: 1000 })

}

**export** **function** **getAddressDetail**(id) {

**return** axios.**get**(`/address/${id}`)

}

接着我们继续上述代码，给 init 方法添加获取地址详情的接口：

methods: {

**async** **init**() {

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

*// 获取查询参数内的 id*

**const** { addressId, cartItemIds } = this.$route.query

*// id 会本地存储，如果查询字符串 id 优先获取，若没有则获取本地存储的 ids*

**const** \_cartItemIds = cartItemIds ? **JSON**.**parse**(cartItemIds) : **JSON**.**parse**(**getLocal**('cartItemIds'))

**setLocal**('cartItemIds', **JSON**.**stringify**(\_cartItemIds))

**const** { data: list } = **await** **getByCartItemIds**({ cartItemIds: \_cartItemIds.**join**(',') })

*// addressId 有的情况下，优先获取 addressI，否则获取默认地址接口*

**const** { data: address } = addressId ? **await** **getAddressDetail**(addressId) : **await** **getDefaultAddress**()

**if** (!address) {

this.$router.**push**({ path: 'address' })

**return**

}

this.cartList = list

this.address = address

**Toast**.**clear**()

},

goTo () {

this.$router.**push**({ path: `address?cartItemIds=${JSON.stringify(this.cartItemIds)}` })

}

}

模板获取数据：

<div class="address-wrap">

<div class="name" @click="goTo">

<span>{{ address.userName }} </span>

<span>{{ address.userPhone }}</span>

</div>

<div class="address">

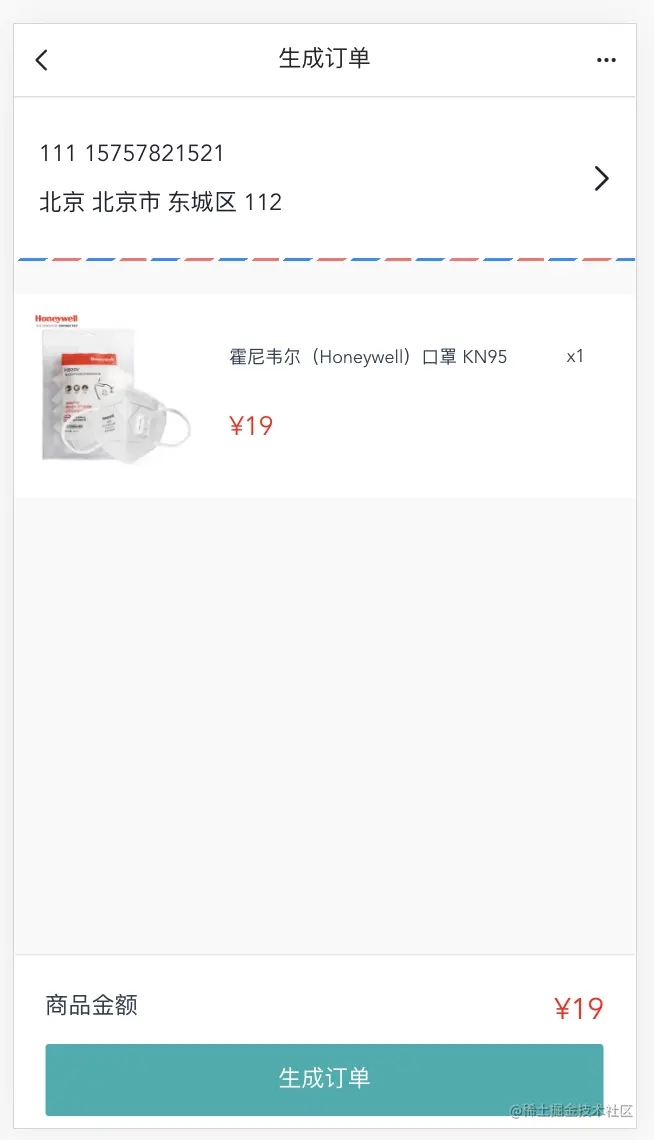
{{ address.provinceName }} {{ address.cityName }} {{ address.regionName }} {{ address.detailAddress }}

</div>

<van-icon class="arrow" name="arrow" />

</div>

我的账号已经有默认数据，所以浏览器显示状态如下：



完整代码如下所示：

<template>

<div class="create-order">

<s-header :title="'生成订单'"></s-header>

<div class="address-wrap">

<div class="name" @click="goTo">

<span>{{ address.userName }} </span>

<span>{{ address.userPhone }}</span>

</div>

<div class="address">

{{ address.provinceName }} {{ address.cityName }} {{ address.regionName }} {{ address.detailAddress }}

</div>

<van-icon class="arrow" name="arrow" />

</div>

<div class="good">

<div class="good-item" v-for="(item, index) in cartList" :key="index">

<div class="good-img"><img :src="`//lmall.xinfeng.site${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>

</div>

</div>

</div>

</div>

<div class="pay-wrap">

<div class="price">

<span>商品金额</span>

<span>¥{{ total }}</span>

</div>

<van-button class="pay-btn" color="#1baeae" type="primary" block>生成订单</van-button>

</div>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

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

import { getDefaultAddress, getAddressDetail } from '../service/address'

import { setLocal, getLocal } from '@/common/js/utils'

import { Toast } from 'vant'

export default {

components: {

sHeader

},

data() {

return {

cartList: [],

address: {}

}

},

mounted() {

this.init()

},

methods: {

async init() {

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

const { addressId, cartItemIds } = this.$route.query

const \_cartItemIds = cartItemIds ? JSON.parse(cartItemIds) : JSON.parse(getLocal('cartItemIds'))

setLocal('cartItemIds', JSON.stringify(\_cartItemIds))

const { data: list } = await getByCartItemIds({ cartItemIds: \_cartItemIds.join(',') })

const { data: address } = addressId ? await getAddressDetail(addressId) : await getDefaultAddress()

if (!address) {

this.$router.push({ path: 'address' })

return

}

this.cartList = list

this.address = address

Toast.clear()

},

goTo() {

this.$router.push({ path: `address?cartItemIds=${JSON.stringify(this.cartItemIds)}` })

}

},

computed: {

total: function() {

let sum = 0

this.cartList.forEach(item => {

sum += item.goodsCount \* item.sellingPrice

})

return sum

}

}

}

</script>

<style lang="less" scoped>

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

.create-order {

background: #f9f9f9;

.address-wrap {

margin-bottom: 20px;

background: #fff;

position: relative;

margin-top: 44px;

font-size: 14px;

padding: 15px;

color: #222333;

.name, .address {

margin: 10px 0;

}

.arrow {

position: absolute;

right: 10px;

top: 50%;

transform: translateY(-50%);

font-size: 20px;

}

&::before {

position: absolute;

right: 0;

bottom: 0;

left: 0;

height: 2px;

background: -webkit-repeating-linear-gradient(135deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background: repeating-linear-gradient(-45deg, #ff6c6c 0, #ff6c6c 20%, transparent 0, transparent 25%, #1989fa 0, #1989fa 45%, transparent 0, transparent 50%);

background-size: 80px;

content: '';

}

}

.good {

margin-bottom: 120px;

}

.good-item {

padding: 10px;

background: #fff;

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;

}

}

}

}

.pay-wrap {

position: fixed;

bottom: 0;

left: 0;

width: 100%;

background: #fff;

padding: 10px 0;

padding-bottom: 50px;

border-top: 1px solid #e9e9e9;

>div {

display: flex;

justify-content: space-between;

padding: 0 5%;

margin: 10px 0;

font-size: 14px;

span:nth-child(2) {

color: red;

font-size: 18px;

}

}

.pay-btn {

position: fixed;

bottom: 7px;

right: 0;

left: 0;

width: 90%;

margin: 0 auto;

}

}

}

</style>

## 总结

本章节我们完成了确认订单页面的制作，涉及到页面内容较多，对数据处理的内容也相对复杂，需要考虑多种逻辑情况，同学们还需仔细研究，也可优化本章节代码，寻找最优的逻辑思路，下一节我们将讲解如何制作地址栏管理页面。