本章节同学们将会学习到：

* 底部导航提取公共组件
* iconfont 添加导航栏字体图标
* 根据不同路由来显示隐藏底部菜单
* 公用头部提取组件
* 接口提取以及接口文档的介绍

## 提取公用底部导航

一个移动端 web 项目中必不可少的内容就是导航，并且导航出现的频率极高，我们若是在每个导航出现的页面复制粘贴，会出现大量的重复工作，这可不是我们开发人员希望看到的情况，所以我们需要进行公共部分代码的抽取和复用。

#### 新建导航组件

第一步，新建导航组件，在 src/components 文件夹下新建导航组件 NavBar.vue ，代码如下：

<template>

<div class="nav-bar">

<ul class="nav-list">

<router-link tag="li" class="nav-list-item active" to="home">

<span>首页</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="category">

<span>分类</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="cart">

<span>购物车</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="user">

<span>我的</span>

</router-link>

</ul>

</div>

</template>

<script>

export default {

}

</script>

<style lang="less" scoped >

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

.nav-bar{

position: fixed;

left: 0;

bottom: 0;

width: 100%;

padding: 5px 0;

z-index: 1000;

background: #fff;

transform: translateZ(0);

-webkit-transform: translateZ(0);

.nav-list {

width: 100%;

.fj();

flex-direction: row;

padding: 0;

.nav-list-item {

display: flex;

flex: 1;

flex-direction: column;

text-align: center;

color: #666;

&.router-link-active {

color: @primary;

}

i {

text-align: center;

font-size: 22px;

}

span{

font-size: 12px;

}

}

}

}

</style>

router-link 是 vue-router 提供的标签，在模板中可以直接使用，用于改变浏览器地址，继而触发视图的修改，在前面的路由篇已经详细说明。

上述代码定义了四个底部导航，分别是「首页」、「分类」、「购物车」、「我的」，当然，项目肯定远不止是这四个页面，只不过这四个页面的 「meta」等级我们会设为 1。

接下来我们需要事先创建好「首页」、「分类」、「购物车」、「我的」这几个页面，在 views 文件夹下分别新建 Home.vue、Category.vue、Cart.vue、User.vue，全部设置成 Vue 基础模板页面，代码如下所示：

// Home.vue

<template>

<div class="home">

<div>我是首页</div>

</div>

</template>

<script>

export default {

name: 'Home'

}

</script>

<style lang="less" scoped>

</style>

// Category.vue

<template>

<div class="category-box">

<div>我是分类页</div>

</div>

</template>

<script>

export default {

name: 'Category'

}

</script>

<style lang="less" scoped>

</style>

// Cart.vue

<template>

<div class="cart-box">

<div>我是购物车</div>

</div>

</template>

<script>

export default {

name: 'Cart'

}

</script>

<style lang="less" scoped>

</style>

// User.vue

<template>

<div class="user-box">

<div>我的</div>

</div>

</template>

<script>

export default {

name: 'User'

}

</script>

<style lang="less" scoped>

</style>

之后再修改我们的路由页面 src/router/index.js，代码如下：

**import** **Vue** **from** 'vue'**import** **VueRouter** **from** 'vue-router'**import** **Home** **from** '../views/Home.vue'

**Vue**.**use**(**VueRouter**)

**const** routes = [

*// 首页我们需要默认空路径重定向到 home 下，避免空页面*

{

path: '/',

name: 'home',

redirect: 'home'

},

{

path: '/home',

name: 'home',

component: **Home**,

meta: {

index: 1

}

},

{

path: '/category',

name: 'category',

component: () => **import**(*/\* webpackChunkName: "category" \*/* '../views/Category.vue'),

meta: {

index: 1

}

},

{

path: '/cart',

name: 'cart',

component: () => **import**(*/\* webpackChunkName: "cart" \*/* '../views/Cart.vue'),

meta: {

index: 1

}

},

{

path: '/user',

name: 'user',

component: () => **import**(*/\* webpackChunkName: "user" \*/* '../views/User.vue'),

meta: {

index: 1

}

},

]

**const** router = **new** **VueRouter**({

routes

})

**export** **default** router

最后打开 App.vue ，引入并注册底部导航组件：

<template>

<div id="app">

<transition :name="transitionName">

<router-view class="router-view" />

</transition>

<nav-bar></nav-bar>

</div>

</template>

<script>

import navBar from '@/components/NavBar'

export default {

data () {

return {

transitionName: 'slide-left'

}

},

components: {

navBar

},

watch: {

$route(to, from) {

// 由主级到次级

if (to.meta.index > from.meta.index) {

this.transitionName = 'slide-left' // 向左滑动

} else if (to.meta.index < from.meta.index) {

// 由次级到主级

this.transitionName = 'slide-right'

} else {

this.transitionName = '' //同级无过渡效果

}

}

}

}

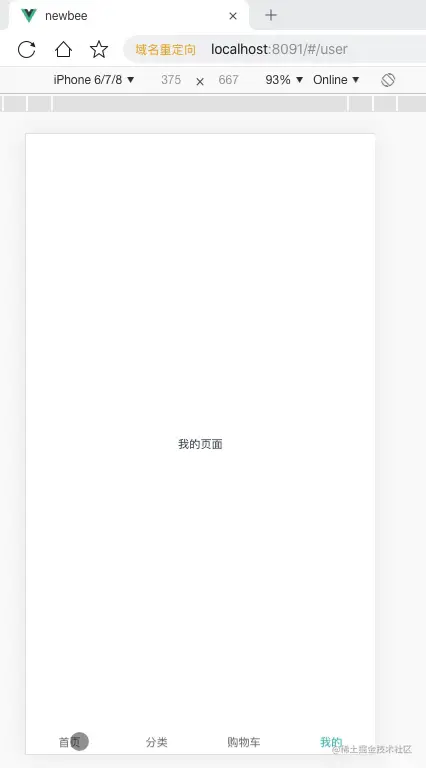
</script>

<style lang="less">

...

</style>

保存之后，来观察浏览器表现情况：



至此，有些人可能会有疑问，若是把 <nav-bar></nav-bar> 放在 App.vue 中，那每个页面都会带上底部导航，这不是我们想要的。比如详情页、购买页等都不需要底部导航。

前面的章节我们讲解过 App.vue 中，watch 属性内可以通过 $route 监听路由的变化，我们可以通过路径判断哪些页面需要底部导航。

首先设置一个变量 isShowNav 用于控制导航栏的显示隐藏，并且在 $route 中判断这个变量是否为 true:

**data**() {

**return** {

transitionName: 'slide-left',

isShowNav: true,

**ShowMenuList**: ['/', '/home', '/category', '/cart', '/user'], *// 该变量为需要导航栏的数组*

}

},watch: {

$route(to, **from**) {

*// 通过 ES6 提供的 includes 属性判断 to.path 是否包含在数组内*

**if** (this.ShowMenuList.**includes**(to.path)) {

this.isShowNav = true

} **else** {

this.isShowNav = false

}

**if** (to.meta.index > **from**.meta.index) {

this.transitionName = 'slide-left' *// 向左滑动*

} **else** **if** (to.meta.index < **from**.meta.index) {

*// 由次级到主级*

this.transitionName = 'slide-right'

} **else** {

this.transitionName = '' *//同级无过渡效果*

}

}

}

利用 isShowNav 控制底部导航栏的显示隐藏：

<template>

<div id="app">

<transition :name="transitionName">

<router-view class="router-view" />

</transition>

<nav-bar v-if="isShowNav"></nav-bar>

</div>

</template>

由于目前设置的页面都是需要导航栏，所以我们需要在 views 下新建一个不需要导航栏的内页 Detail.vue ，并且在 router/index.js 添加路由：

*// router/index.js*

{

path: '/detail',

name: 'detail',

component: () => **import**(*/\* webpackChunkName: "detail" \*/* '../views/Detail.vue'),

meta: {

index: 2

}

},

重启项目并访问，观察浏览器视图的变化如下：



#### 公用头部组件提取

提取了底部导航栏，接下来需要把页面的头部也提取出来。

一个头部组件需要传入 title 信息作为头部文字显示，新建 src/components/SimpleHeader.vue 在代码中看注释：

<template>

<header class="simple-header">

<i class="nbicon nbfanhui" @click="goBack"></i>

<div class="simple-header-title">{{ title }}</div>

<i class="nbicon nbmore"></i>

</header>

</template>

<script>

export default {

// 外部传入的 title 属性，可以定义类型和默认值

props: {

title: {

type: String,

default: ''

}

},

methods: {

// 点击头部组件右侧返回按钮，让路由往回切一页

goBack() {

this.$router.go(-1)

}

}

}

</script>

<style lang="less" scoped>

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

.simple-header {

position: fixed;

top: 0;

left: 0;

z-index: 10000;

.fj();

.wh(100%, 44px);

line-height: 44px;

padding: 0 10px;

.boxSizing();

color: #252525;

background: #fff;

border-bottom: 1px solid #dcdcdc;

.simple-header-title {

font-size: 14px;

}

}

</style>

在详情页中尝试引入 SimpleHeader 组件，如下所示：

<template>

<div>

<s-header :title="'详情'"></s-header>

</div>

</template>

<script>

import sHeader from '@/components/SimpleHeader'

export default {

components: {

sHeader

}

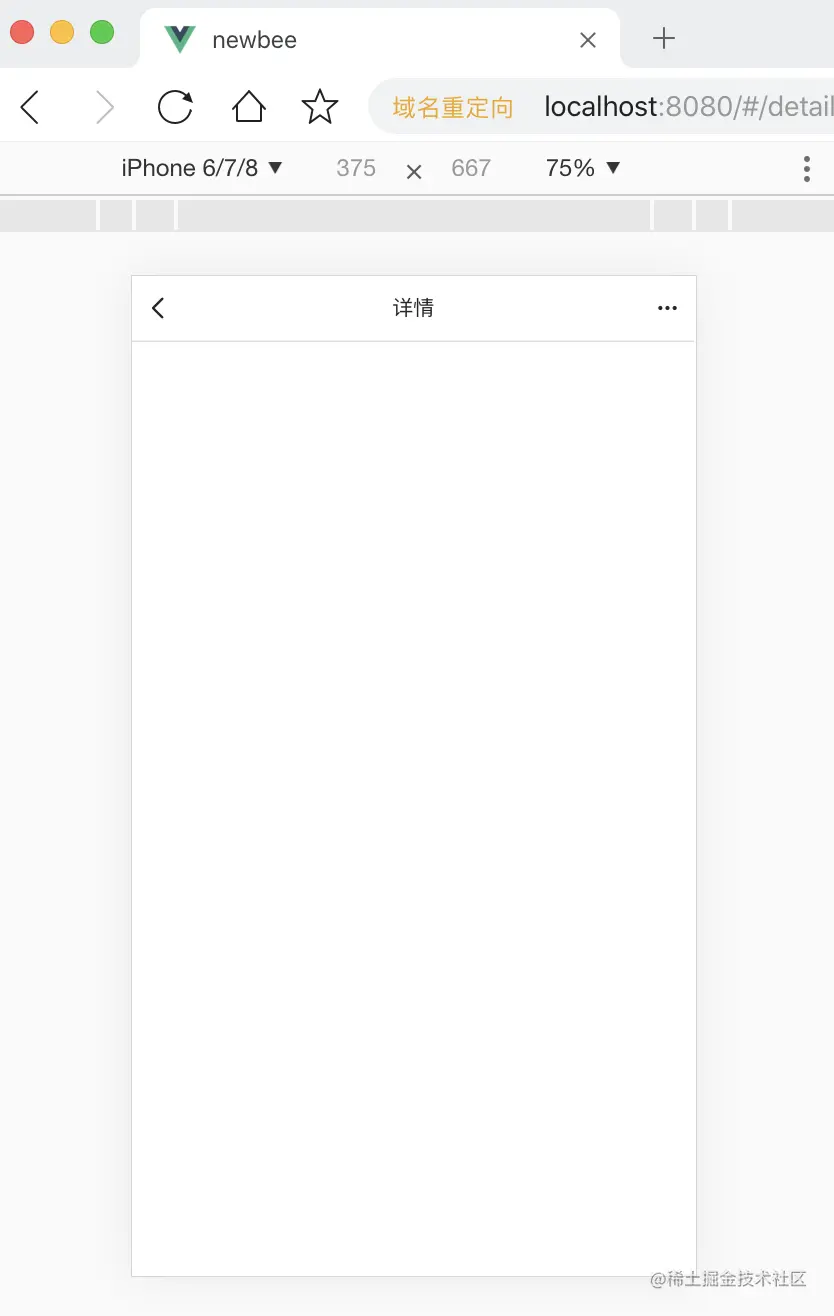
}

</script>

<style>

</style>

访问浏览器显示如下，头部信息的设置完成。



#### 添加 iconfont 字体图标

首先我们需要去之前在搭建 H5 开发环境时，创建的 iconfont 项目下添加几个图标。

新蜂商城项目选择了下面几个图标：



大家可以根据自己的喜好选择，并且我将项目的样式前缀改成了 nbicon，修改完之后大家不要忘记重新生成新的静态资源链接，将新的链接重新覆盖 public/index.html：

**<!DOCTYPE html>**<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no">

<link rel="icon" href="<%= BASE\_URL %>favicon.ico">

<title><%= htmlWebpackPlugin.options.title %></title>

<link rel="stylesheet" href="//at.alicdn.com/t/font\_1623819\_3g3arzgtlmk.css">

</head>

<body>

<noscript>

<strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled. Please enable it to continue.</strong>

</noscript>

<div id="app"></div>

*<!-- built files will be auto injected -->*

</body></html>

别忘记了，我们还没有将 icon 添加到页面中呢，修改 components/NavBar.vue ，代码如下：

<template>

<div class="nav-bar">

<ul class="nav-list">

<router-link tag="li" class="nav-list-item active" to="home">

<i class="nbicon nblvsefenkaicankaoxianban-1"></i>

<span>首页</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="category">

<i class="nbicon nbfenlei"></i>

<span>分类</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="cart">

<i class="nbicon nbgouwuche"></i>

<span>购物车</span>

</router-link>

<router-link tag="li" class="nav-list-item" to="user">

<i class="nbicon nblvsefenkaicankaoxianban-"></i>

<span>我的</span>

</router-link>

</ul>

</div>

</template>

<script>

export default {

}

</script>

<style lang="less" scoped >

...

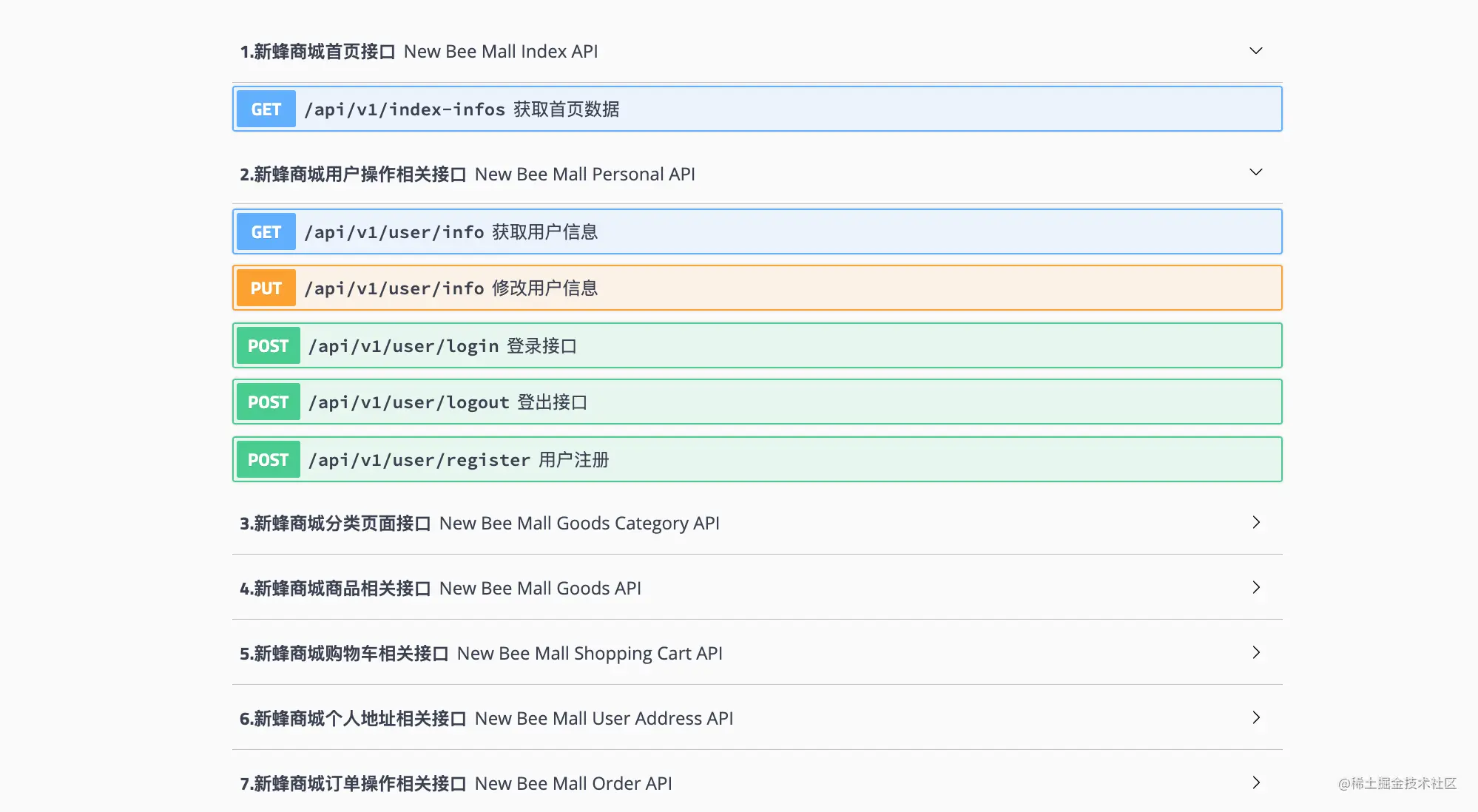
</style>

保存之后，浏览器效果如下：



#### 接口文档及请求地址封装

为了保证项目尽量靠近真实开发场景，这里为大家提供[接口文档](https://link.juejin.cn/?target=http://backend-api-01.newbee.ltd/swagger-ui.html" \o "http://backend-api-01.newbee.ltd/swagger-ui.html" \t "/home/erfenjiao/文档\\x/_blank)，项目中所有的接口都已经在文档中体现，这与真实开发项目时的内容基本一致。



我们还需将需要用到的接口地址进行二次封装，以接下来要接入的登录鉴权接口来举例。

首先在项目 src 目录下新建 service 文件夹，在 service 文件夹下新建 user.js 文件，代码如下：

*// user.js***import** axios **from** '../utils/axios' *// 二次封装的 axios*

*// 获取用户信息***export** **function** **getUserInfo**() {

**return** axios.**get**('/user/info');

}

*// 登录***export** **function** **login**(params) {

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

}

*// 登出***export** **function** **logout**() {

**return** axios.**post**('/user/logout')

}

*// 注册***export** **function** **register**(params) {

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

}

统一将用户信息鉴权相关的接口放到 user.js 文件下，这样做的目的是之后有修改到接口地址的情况，不需要在每个使用到该接口的地方都做修改，只需要修改 user.js 文件下的方法。

写法不止上面这一种，同学们可以利用好 ES6 给我们提供的模块化能力，将代码封装的更加健壮。

## 总结

本章节我们学习了导航栏的创建、字体图标的筛选及引入、接口的二次封装。之所以先实现导航栏及相关交互，是因为这个模块承载着整个项目的骨架。