一、常用组件

在上一个章节中讲解了封装请求数据的模块，在此处请求轮播图的数据

1.首页轮播图数据的请求以及渲染

1.1 轮播图数据的请求 pages/home/home.js

```
import { request } from '../../../utils/index.js'
Page({

  /**
  * 页面的初始数据
  */
  data: {
  bannerlist: []
  },

```

```
  /**
  * 生命周期函数--监听页面加载
  */
  onLoad: function (options) {
  request('/api/pro/banner').then(data => {
  console.log(data)
  // 微信小程序修改数据的方式
  this.setData({
  bannerlist: data.data
  })
  })
  },
 })
```

2 使用组件 - 视图容器 - swiper

滑块视图容器。其中只可放置swiper-item组件，否则会导致未定义的行为。
属性表如下

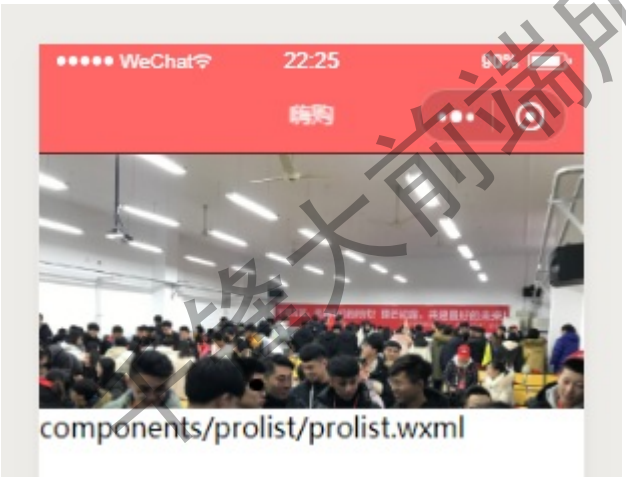
属性	类型	默认值	必填	说明	最低版本
indicator-dots	boolean	false	否	是否显示面板指示点	1.0.0
indicator-color	color	rgba(0, 0, 0, .3)	否	指示点颜色	1.1.0
indicator-active-color	color	#000000	否	当前选中的指示点颜色	1.1.0
autoplay	boolean	false	否	是否自动切换	1.0.0
current	num	0	否	当前所在滑块的 index	1.0.

	ber				0
interval	number	5000	否	自动切换时间间隔	1.0.0
duration	number	500	否	滑动动画时长	1.0.0
circular	boolean	false	否	是否采用衔接滑动	1.0.0
vertical	boolean	false	否	滑动方向是否为纵向	1.0.0
previous-margin	string	“0px”	否	前边距，可用于露出前一项的一小部分，接受 px 和 rpx 值	1.9.0
next-margin	string	“0px”	否	后边距，可用于露出后一项的一小部分，接受 px 和 rpx 值	1.9.0
display-multiple-items	number	1	否	同时显示的滑块数量	1.9.0
skip-hidden-item-layout	boolean	false	否	是否跳过未显示的滑块布局，设为 true 可优化复杂情况下的滑动性能，但会丢失隐藏状态滑块的布局信息	1.9.0
easing-function	string	“default”	否	指定 swiper 切换缓动动画类型	2.6.5
bindchange	eventhandle		否	current 改变时会触发 change 事件，event.detail = {current, source}	1.0.0
bindtransi	eventh		否	swiper-item 的位置发生改变时会触发 transition	2.4.

tion	and le			事件，event.detail = {dx: dx, dy: dy}	3
bindanimationfinish	eventhandle		否	动画结束时会触发 animationfinish 事件，event.detail 同上	1.9.0

在pages/home/home.wxml文件中输入如下代码查看效果

```
<!--pages/home/home.wxml-->
<swiper
 indicator-dots="{{true}}" autoplay="{{true}}" circular="{{true}}"
 duration="{{500}}">
 <block wx:for="{{bannerlist}}" wx:key="index">
 <swiper-item >
 <image src="{{'http://daxun.kuboy.top' + item}}"></image>
 </swiper-item>
 </block>
</swiper>
<prolist></prolist>
```



二、自定义组件 - 产品列表

1. 自定义组件的布局

components/prolist/prolist.wxml

```
<view class="prolist">
```

```
  <view class="proitem">
  <view class="itemimg">
  <image class="img" src=""></image>
  </view>
  <view class="iteminfo">
  <view class="title">
  产品名称
  </view>
  <view class="price">
  ￥199
  </view>
  </view>
  </view>
 </view>
```

2. 自定义组件的样式

components/prolist/prolist.wxss

```
/* components/prolist/prolist.wxss */
.prolist .proitem{
  width: 100%;
  display: flex;
  height: 100px;
  box-sizing: border-box;
  border-bottom: 1px solid #ccc;
}

.prolist .proitem .itemimg{
  width: 100px;
  height: 100px;
  padding: 5px;
}

.prolist .proitem .itemimg .img{
  width: 90px;
  height: 90px;
  box-sizing: border-box;
  border: 1px solid #ccc;
 }
```

```
22.
23. .prolist .proitem .iteminfo {
24.  padding: 3px;
25. }
26.
27. .prolist .proitem .iteminfo .title{
28.  font-size: 18px;
29.  font-weight: bold;
30. }
31.
32. .prolist .proitem .iteminfo .price{
33.  font-size: 12px;
34. }
```

3. 首页请求数据，并且传递给子组件

pages/home/home.js

```
1. import { request } from './../../utils/index.js'
2. Page({
3.
4.  /**
5.  * 页面的初始数据
6.  */
7.  data: {
8.  prolist: []
```

```
    prolist: []
  },

  /**
  * 生命周期函数--监听页面加载
  */
  onLoad: function (options) {
  request('/api/pro').then(data => {
  console.log(data)
  // 微信小程序修改数据的方式
  this.setData({
  prolist: data.data
  })
  })
  },
})
```

pages/home/home.wxml

```
<prolist prolist="{{prolist}}"></prolist>
```

4. 子组件接收数据

components/prolist/prolist.js

```
Component({
  /**
  * 组件的属性列表
  */
  properties: {
  prolist: Array
  },
})
```

5. 子组件渲染数据

components/prolist/prolist.wxml

```
<view class="prolist">
  <view class="proitem" wx:for="{{prolist}}" wx:key="item.proid">
  <view class="itemimg">
  <image class="img" src="{{item.proimg}}"></image>
```

```
5.    </view>
6.    <view class="iteminfo">
7.    <view class="title">
8.    {{item.proname}}
9.    </view>
10.   <view class="price">
11.   ￥{{item.price}}
12.   </view>
13.   </view>
14.   </view>
15.  </view>
```

三、实现下拉刷新上拉加载

1. 开启首页的下拉刷新功能

pages/home/home.json

```
{
 "usingComponents": {
 "prolist": "/components/prolist/prolist"
 },
 "enablePullDownRefresh": true,
 "backgroundColor": "#efefef",
 "backgroundTextStyle": "dark"
}
```

2. 完善相关的下拉刷新函数

pages/home/home.js

```
// pages/home/home.js
import { request } from '../../../utils/index.js'
Page({

 /**
 * 页面的初始数据
 */
 data: {
 bannerlist: [],
 prolist: [],
 pageCode: 1 // 页码
 },

 /**
 * 生命周期函数--监听页面加载
 */
 onLoad: function (options) {
 request('/api/pro/banner').then(data => {
 console.log(data)
 this.setData({
 bannerlist: data.data
 })
 })

 request('/api/pro').then(data => {
 console.log(data)
```

```
27.   this.setData({
28.   prolist: data.data
29.   })
30.   })
31.   },
32.
33.   /**
34.   * 页面相关事件处理函数--监听用户下拉动作
35.   */
36.   onPullDownRefresh: function () {
37.   request('/api/pro').then(data => {
38.   console.log(data)
39.   this.setData({
40.   prolist: data.data,
41.   pageCode: 1
42.   })
43.   })
44.   },
45.
46.   /**
47.   * 页面上拉触底事件的处理函数
48.   */
49.   onReachBottom: function () {
50.   let num = this.data.pageCode;
51.   let prolist = this.data.prolist
52.   num++;
53.   console.log(num)
54.   request('/api/pro', {
55.   pageCode: num
56.   }).then(data => {
57.   // 此处注意临界值的变化 -- 没有数据
58.   this.setData({
59.   prolist: [...prolist, ...data.data],
60.   pageCode: num
61.   })
62.   })
63.   }
64.  })
```

上拉下拉测试即可

四、近同项部功能实现

四、返回顶部功能实现

在首页中设置一个固定定位的按钮，然后绑定点击事件，绑定事件使用 bindtap，然后调用小程序提供的api即可返回

```
// pages/home/home.wxml
<view class="backtop" bindtap="backtop"> ↑ </view>

// pages/home/home.wxss
.backtop {
  position: fixed;
  bottom: 10px;
  right: 8px;
  border-radius: 50%;
  width: 30px;
  height: 30px;
  background-color: rgba(0, 0, 0, 0.5);
  font-size: 18px;
  text-align: center;
  line-height: 30px;
}

// pages/home/home.js
Page({
  /**
  * 自定义函数
  */
  backtop: function () {
  // 小程序api 的界面 - 滚动
  wx.pageScrollTo({
  scrollTop: 0,
  duration: 300
  })
  }
})
```

五、实现点击商品列表进入产品的详情页面

1. 构建详情页面

app.json

```
"pages": [
 "pages/detail/detail"
],
```

2.声明式导航跳转

使用小程序 组件-导航-navigator

页面链接。

属性	类型	默认值	必填	说明	最低版本
target	string	self	否	在哪个目标上发生跳转，默认当前小程序	2.0.7
url	string		否	当前小程序内的跳转链接	1.0.0
open-type	string	navigate	否	跳转方式	1.0.0
delta	number	1	否	当 open-type 为 ‘navigateBack’ 时有效，表示回退的层数	1.0.0
app-id	string		否	当 `target=”miniProgram”` 时有效，要打开的小程序 appId	2.0.7
path	string		否	当 `target=”miniProgram”` 时有效，打开的页面路径，如果为空则打开首页	2.0.7
extra-d	ob			当 `target=”miniProgram”` 时有效，需要传递给目标小程	2.

extra-data	object		否	序的数据，目标小程序可在 App.onLaunch() ， App.onShow() 中获取到这份数据。详情	0.7
version	string	release	否	当 target=”miniProgram” 时有效，要打开的小程序版本	2.0.7
hover-class	string	navigator-hover	否	指定点击时的样式类，当 hover-class=”none” 时，没有点击态效果	1.0.0
hover-stop-propagation	boolean	false	否	指定是否阻止本节点的祖先节点出现点击态	1.5.0
hover-start-time	number	50	否	按住后多久出现点击态，单位毫秒	1.0.0
hover-stay-time	number	600	否	手指松开后点击态保留时间，单位毫秒	1.0.0
bindsuccess	string		否	当 target=”miniProgram” 时有效，跳转小程序成功	2.0.7
bindfail	string		否	当 target=”miniProgram” 时有效，跳转小程序失败	2.0.7
bindcomplete	string		否	当 target=”miniProgram” 时有效，跳转小程序完成	2.0.7

open-type 的合法值 — 在编程式导航中详细讲解

值	说明	最低版本
navigate	对应 wx.navigateTo 或 wx.navigateToMiniProgram 的功能	
redirect	对应 wx.redirectTo 的功能	

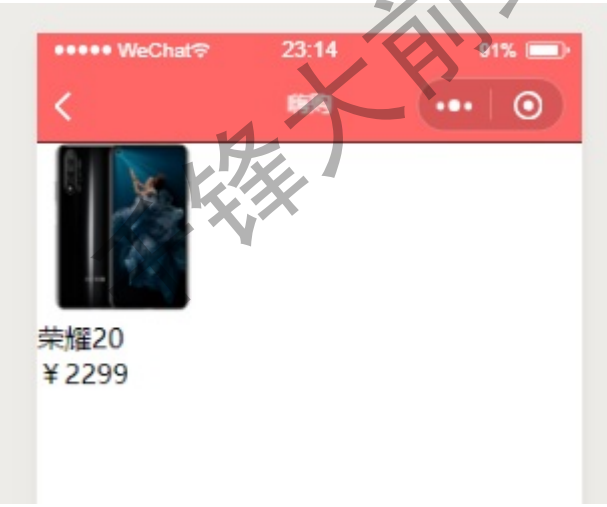
switchTab	对应 wx.switchTab 的功能	
reLaunch	对应 wx.reLaunch 的功能	1.1.0
navigateBack	对应 wx.navigateBack 的功能	1.1.0
exit	退出小程序，target="miniProgram" 时生效	2.1.0

```
// components/prolist/prolist.wxml
<view class="prolist">
  <navigator url="{{'/pages/detail/detail?proid=' + item.proid}}" wx:for="{{prolist}}" wx:key="item.proid">
  <view class="proitem" >
  <view class="itemimg">
  <image class="img" src="{{item.proimg}}"></image>
  </view>
  <view class="iteminfo">
  <view class="title">
  {{item.proname}}
  </view>
  <view class="price">
  ￥{{item.price}}
  </view>
  </view>
  </view>
  </navigator>
</view>
```

3. 详情页面接收数据并且渲染数据

```
// pages/detail/detail.js
import { request } from '../../../utils/index.js';
Page({

  /**
  * 页面的初始数据
  */
  data: {
  proid: '',
  proname: '',
  proimg: ''
  price: 0
```

```
  },

  /**
  * 生命周期函数--监听页面加载
  */
  onLoad: function (options) {
  // options为接收的参数
  const { proid } = options
  request('/api/pro/detail?proid=' + proid).then(data => {
  console.log(data.data)
  const { proid, proname, price, proimg} = data.data
  this.setData({
  proid, proname, price, proimg
  })
  })
  }
 })

 // pages/detail/detail.wxml

 <image src="{{proimg}}" style="width: 100px;height: 100px;"></image>
 <view>{{proname}}</view>
 <view>￥{{price}}</view>
```



点击不同的产品测试即可

4. 编程式导航渲染

使用小程序提供的api实现编程式路由的跳转

使用小程序提供的api实现编程式路由的跳转

wx.switchTab(Object object)

跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面

wx.reLaunch(Object object)

关闭所有页面，打开到应用内的某个页面

wx.redirectTo(Object object)

关闭当前页面，跳转到应用内的某个页面。但是不允许跳转到 tabbar 页面

wx.navigateTo(Object object)

保留当前页面，跳转到应用内的某个页面。但是不能跳到 tabbar 页面。使用 wx.navigateBack 可以返回到原页面。小程序中页面栈最多十层

wx.navigateBack(Object object)

关闭当前页面，返回上一页面或多级页面。可通过 getCurrentPages 获取当前的页面栈，决定需要返回几层

> *小程序传递数据使用 data-params形式，可以在事件中根据event获取该参数*

```
// components/prolist/prolist.wxml
<view class="prolist">
 <view class="proitem" bindtap="toDetail" data-proid="{{item.proid}}" wx:for="{{prolist}}" wx:key="item.proid">
 <view class="itemimg">
 <image class="img" src="{{item.proimg}}"></image>
 </view>
 <view class="iteminfo">
 <view class="title">
 {{item.proname}}
 </view>
 <view class="price">
 ￥{{item.price}}
 </view>
 </view>
 </view>
</view>

// components/prolist/prolist.js
```

```
// components/prolist/prolist.js
Component({
  /**
  * 组件的属性列表
  */
  properties: {
  prolist: Array
  },

  /**
  * 组件的初始数据
  */
  data: {

  },

  /**
  * 组件的方法列表
  */
  methods: {
  toDetail (event) {
  const { proid } = event.currentTarget.dataset
  wx.navigateTo({
  url: '/pages/detail/detail?proid=' + proid
  })
  }
  }
})
```