第 4 章　莫凡商城首页静态布局设计

课时内容	视图容器组件和基础内容组件	课时	2
教学目标	了解首页功能与所需组件 熟悉视图容器组件和基础内容组件 掌握搜索区域布局与海报轮播功能的实现		
教学重点	图片组件及图片 API、导航组件和导航 API 莫凡商城首页静态布局的方法		
教学难点	视图容器组件和基础内容组件		
教学设计	1. 教学思路：讲解莫凡商城首页功能与所需组件；通过多媒体演示和实机操作讲解视图容器组件和基础内容组件的使用；通过实战实现搜索区域布局与海报轮播功能。 2. 教学手段：多媒体+计算机 3. 教学资料：教材、多媒体课件		

教学内容

莫凡商城首页包含 3 个部分的功能：第 1 部分是用来搜索莫凡商城商品的搜索区域；第 2 部分是通过海报轮播效果显示的轮播广告；第 3 部分用来展示图书商品，每个区域展示 3 本图书，可以通过单击“查看更多”按钮找到更多的图书商品，如图所示。

首页效果

4.1 首页需求分析与知识点

莫凡商城首页包含搜索、海报轮播、展示图书商品、查看更多图书商品功能，它的布局设计需要用到视图容器组件，如 view 视图容器组件、swiper 滑块视图容器组件、text 文本组件、image 图片组件等。我们会详细介绍这些组件及其相关组件的使用方法，海报轮播效果使用 swiper 滑块视图容器组件来实现；单击搜索区域和查看更多图书商品时会进行页面跳转，会用到导航组件和导航 API；界面的样式需要用 WXSS 样式来渲染，进行页面美化布局，最终才能完成莫凡商城首页的静态布局设计。

4.2 视图容器组件在首页中的应用

视图容器组件是用来进行页面布局的，不同的视图容器组件可以用来实现不同的布局效果，view 视图容

器组件是基本的容器组件，scroll-view 是可滚动视图容器组件，swiper 是滑块视图容器组件，movable-view 是可移动视图容器组件，cover-view 是覆盖原生组件的视图容器组件。

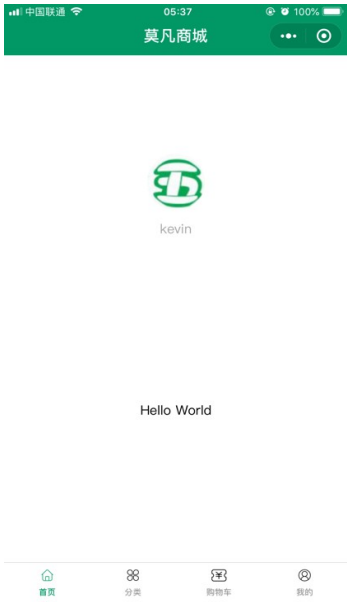
4.2.1 view 视图容器组件

view 视图容器组件是 WXML 界面布局的基础组件，也是最常用的界面布局组件，它的使用和 HTML 里的 DIV 功能类似。view 视图容器组件有自己的属性，如表所示。

view 的属性

属性	类型	默认值	说明
hover-class	string	none	指定按下去的样式类。当 hover-class="none" 时，表示没有单击态效果
hover-stop-propagation	boolean	false	指定是否阻止本节点的祖先节点出现单击态
hover-start-time	number	50	指定按住后多久出现单击态，单位为 ms
hover-stay-time	number	400	指定手指松开后单击态的保留时间，单位为 ms

案例：在莫凡商城 index.wxml 文件页面里，输出“Hello World”文字、头像、昵称可以使用 view 布局，渲染出界面内容，如图 4.2 所示。



view 布局

具体代码如下。

```
<view class="container">
  <view class="userinfo">
                <image bindtap="bindViewTap" class="userinfo-avatar" src="https://wx.qlogo.cn/mmopen/vi_32/ Q0j4TwGTfTI249Dbdib9mqaKWK29vWLp2KlPHrMO0bwmOgLUE9T86XOG9kRx9PtBMRic4HFwqeHbUIK5IDWzvwPA/132" mode="cover"></image>
    <text class="userinfo-nickname">kevin</text>
  </view>
  <view class="usermotto">
    <text class="user-motto">Hello World</text>
  </view>
</view>
```

4.2.2 scroll-view 可滚动视图容器组件

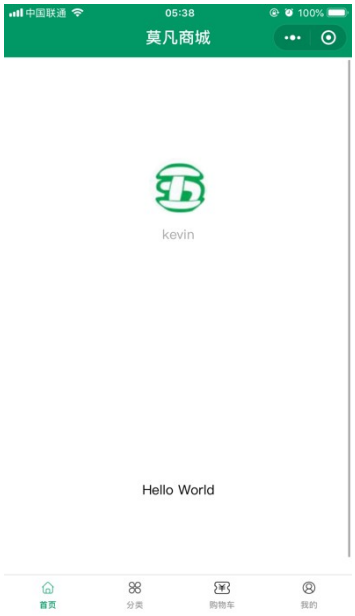
scroll-view 是可滚动视图容器组件，允许视图容器中的内容进行横向滚动或者纵向滚动，类似于浏览器的水平滚动条和垂直滚动条，可以在有限的显示窗口中，通过滚动的方式显示更多的内容。scroll-view 的属性如表所示。

scroll-view 的属性

属性	类型	默认值	说明
scroll-x	Boolean	false	允许横向滚动
scroll-y	Boolean	false	允许纵向滚动
upper-threshold	Number	50	指定距顶部/左边多远时，触发 scrolltoupper 事件，单位为像素
lower-threshold	Number	50	指定距底部/右边多远时，触发 scrolltolower 事件，单位为像素
scroll-top	Number		设置竖向滚动条位置
scroll-left	Number		设置横向滚动条位置
scroll-into-view	String		值应为某子元素 id，则滚动到该元素，元素顶部对齐滚动区域顶部
scroll-with-animation	Boolean	false	在设置滚动条位置时使用动画过渡
enable-back-to-top	Boolean	false	iOS 单击顶部状态栏；安卓双击标题栏时，滚动条返回顶部
bindscrolltoupper	eventHandle		滚动到顶部/左边，会触发 scrolltoupper 事件
bindscrolltolower	eventHandle		滚动到底部/右边，会触发 scrolltolower 事件
bindscroll	eventHandle		滚动时触发，event.detail = {scrollLeft, scrollTop, scrollHeight, scrollWidth, deltaX, deltaY}

1. 纵向滚动

要实现内容纵向滚动，需要给<scroll-view/>一个固定高度，在滑动的时候就会出现纵向的滚动条。在莫凡商城 index.wxml 页面文件里增加纵向滚动条，如图所示。



纵向滚动

具体代码如下。

```
    <view class="container">
    <scroll-view scroll-y="true" style="height: 200px;" bindscrolltoupper="upper" bindscrolltolower="lower">
      <view class="userinfo">
                    <image bindtap="bindViewTap" class="userinfo-avatar" src="https://wx.qlogo.cn/mmopen/vi_32/
Q0j4TwGTfTI249Dbdib9mqaKWK29vWLp2KIPHrMO0bwmOgLUE9T86XOG9kRx9PtBMRic4HFwqeHbUIK5IDWzvwPA/132"
mode="cover"></image>
          <text class="userinfo-nickname">kevin</text>
      </view>
        </scroll-view>
      <view class="usermotto">
        <text class="user-motto">Hello World</text>
      </view>

    </view>
```

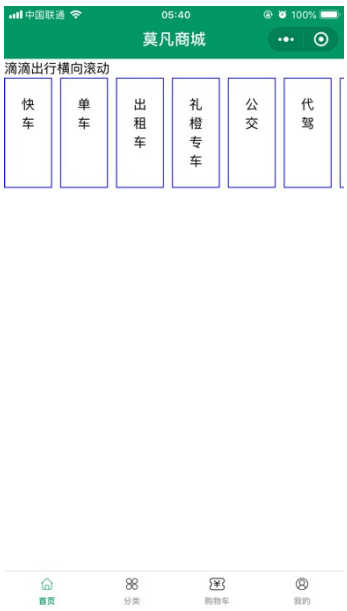
2. 横向滚动

滴滴出行 App 在地图的上方显示可以打车的类别导航，有快车、单车、出租车、礼橙专车、公交、代驾、豪华车、安全须知、小桔租车、顺风车、关爱出行、车生活、金融服务，这些导航在一屏里无法完全显示出来，需要通过向左滑动和向右滑动显示完整的导航内容，单击相应的导航可以看到对应的内容，这时就可以

采用 scroll-view 来实现这些导航的横向滚动，如图所示。

滴滴出行导航

下面模拟滴滴出行横向滚动效果，可以向左滑动和向右滑动，效果如图所示。



模拟滴滴出行横向滚动效果

在 WXML 文件里使用 scroll-view 进行布局，设置 scroll-x="true"横向滚动，具体代码如下。

```
<view class="section">
  <view class="section__title">滴滴出行横向滚动</view>
   <scroll-view scroll-x="true" style="width: 100%;">
    <view style="display:flex;flex-direction:row">
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">快车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">单车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">出租车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">礼橙专车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">公交</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">代驾</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">豪华车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">安全须知</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">小桔租车</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">顺风车</view>
       <view style="margin-right:10px;border:1px solid blue;padding:20px;">关爱出行</view>
      <view style="margin-right:10px;border:1px solid blue;padding:20px;">车生活</view>
```

```
        <view style="margin-right:10px;border:1px solid blue;padding:20px;">金融服务</view>
      </view>
    </scroll-view>
  </view>
```

注意：

（1）基础库 2.4.0 以下不支持嵌套 textarea（多行输入框组件）、map（地图组件）、canvas（画布）和 video（视频）组件；

（2）scroll-into-view 的优先级高于 scroll-top。

（3）在滚动 scroll-view 时会阻止页面回弹，所以在 scroll-view 中滚动，是无法触发下拉刷新 onPullDownRefresh 的；若要使用下拉刷新，需要使用页面的滚动，而不是 scroll-view。

4.2.3 swiper 滑块视图容器组件

swiper 滑块视图容器组件是经常会用到的组件，它可以实现海报轮播效果或者多种登录方式（账号密码登录、手机号快捷登录）之间的切换，可以用来在指定区域内切换不同内容的显示，它的属性如表所示。

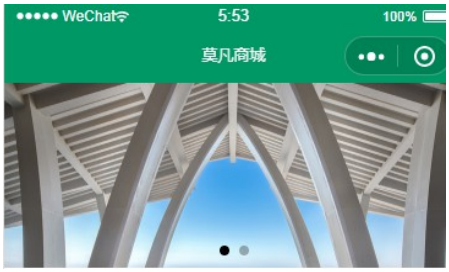
swiper 的属性

属性	类型	默认值	说明
indicator-dots	boolean	false	设置是否显示面板指示点
indicator-color	color	rgba（0, 0, 0, .3）	设置指示点颜色
indicator-active-color	color	#000000	设置当前选中的指示点颜色
autoplay	boolean	false	设置是否自动切换
current	number	0	设置当前所在页面的 index
interval	number	5000	设置自动切换时间间隔
duration	number	500	设置滑动动画时长
circular	boolean	false	设置是否采用衔接滑动
vertical	boolean	false	设置滑动方向是否为纵向
previous-margin	string	"0px"	设置前边距，可用于露出前一项的一小部分，接受 px 和 rpx 值
next-margin	string	"0px"	设置后边距，可用于露出后一项的一小部分，接受 px 和 rpx 值
display-multiple-items	number	1	设置同时显示的滑块数量
skip-hidden-item-layout	boolean	false	设置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，但会丢失隐藏状态下滑块的布局信息
easing-function	string	"default"	指定 swiper 切换缓动动画类型
bindchange	eventHandle		current 改变时会触发 change 事件，event.detail = {current: current}
bindtransition	eventHandle		swiper-item 的位置发生改变时会触发 transition 事件，event.detail = {dx: dx, dy: dy}
bindanimationfinish	eventHandle		动画结束时会触发 animationfinish 事件，event.detail 同上

在 swiper 滑块视图容器组件里，嵌套有 swiper-item 组件，它用来显示不同页签的内容，一个 swiper 滑块视图容器组件里可以有多个 swiper-item 组件，来显示多个区域的内容，以实现海报轮播效果和页签切换效果。

1. 海报轮播效果

海报轮播效果常用来展示商品图片信息或者广告信息。要在有限的区域内展示更多的内容，只能通过轮播的方式动态显示这些内容。海报轮播是网站和 App 都会采用的一种布局方式，如图所示。



海报 1

海报 2

（1）在 WXML 文件里进行海报轮播区域的布局，采用 swiper 滑块视图容器组件进行布局，具体代码如下。

```
<view class="haibao">
  <swiper indicator-dots="{{indicatorDots}}" autoplay="{{autoplay}}" interval="{{interval}}" duration="{{duration}}">
    <block wx:for="{{imgUrls}}">
      <swiper-item>
        <image src="{{item}}" class="silde-image" style="width:100%"></image>
      </swiper-item>
    </block>
  </swiper>
</view>
```

（2）在 JS 文件里，提供海报轮播的图片、是否自动播放、轮播的时长等数据，通过数据绑定的方式渲染到页面上，具体代码如下。

```
Page({
  data: {
    indicatorDots: true,
    autoplay: true,
    interval: 5000,
    duration: 1000,
    imgUrls: [
      "../images/haibao/11.jpg", "../images/haibao/22.jpg"
    ]
  }
})
```

设置 autoplay 等于 true 就可以自动进行海报轮播了，设置 indicatorDots 等于 true 则代表面板显示指示点，同时可以设置 interval 自动切换时长，duration 滑动动画时长。

使用 indicator-color 属性来设置指示点颜色，使用 indicator-active-color 属性来设置当前选中的指示点颜色，这样就可以根据自己的需求来设计更好的海报轮播效果。

2. 页签切换效果

swiper 滑块视图容器组件除了可以实现海报轮播效果，还可以实现页签切换效果。它有一个 current 属性，表示当前所在页面的 index，根据 index 值来显示不同的页面，常用于多种方式的登录或者多种页签导航之间的切换，如图所示。

账号密码登录　　手机快捷登录

（1）进入到 WXML 文件里，进行账号密码登录和手机快捷登录的界面布局设计，具体代码如下。

```
<view class="content">
  <view class="loginTitle">
    <view class="{{currentTab==0?'select':'default'}}" data-current="0" bindtap="switchNav">账号密码登录</view>
    <view class="{{currentTab==1?'select':'default'}}" data-current="1" bindtap="switchNav">手机快捷登录</view>
  </view>
  <view class="hr"></view>
  <swiper current="{{currentTab}}"style="height:120px">
    <swiper-item>
      <view style="margin:0 auto;border:1px solid #cccccc;width:99%;height:100px;">
        账号密码登录区域内容
      </view>
    </swiper-item>
    <swiper-item>
     <view style="margin:0 auto;border:1px solid #cccccc;width:99%;height:100px;">
        手机快捷登录区域内容
      </view>
    </swiper-item>
  </swiper>
</view>
```

（2）进入到 WXSS 文件里，给页面文件添加样式，具体代码如下。

```
.loginTitle{
    display: flex;
    flex-direction: row;
    width: 100%;
}
.select{
    font-size:12px;
    color: red;
    width: 50%;
    text-align: center;
    height: 45px;
    line-height: 45px;
    border-bottom:5rpx solid red;
}
.default{
    font-size:12px;
    margin: 0 auto;
    padding: 15px;
}
.hr{
    border: 1px solid #cccccc;
    opacity: 0.2;
}
```

（3）进入到 JS 文件里，提供当前面板的索引值，提供页签切换函数，具体代码如下。

```
Page({
  data: {
    currentTab: 0
  },
  switchNav: function (e) {
    var page = this;
    if (this.data.currentTab == e.target.dataset.current) {
      return false;
    } else {
      page.setData({ currentTab: e.target.dataset.current });
    }
  }
})
```

这样就可以实现在两种登录状态下的页签切换效果，页签切换时，页签的标题呈现为选中的状态，同时对应的内容也跟着进行切换。

4.2.4 movable-view 可移动视图容器组件

movable-view 是一个可移动视图容器组件，在页面中可以做拖曳滑动，在使用这个组件时，需要先定义可移动区域 movable-area，然后定义直接子节点 movable-view，否则不能移动。movable-area 要设置 width 和 height 属性，不设置时默认为 10 px。movable-view 也要设置 width 和 height 属性，不设置时默认为 10 px, movable-view 默认为绝对定位，top 和 left 属性为 0 px。movable-view 可移动视图容器的属性如表所示。

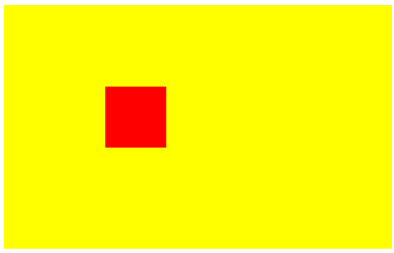
movable-view 的属性

属性	类型	默认值	说明
direction	string	none	设置 movable-view 的移动方向，属性值有 all、vertical、horizontal、none
inertia	boolean	false	设置 movable-view 是否带有惯性
out-of-bounds	boolean	false	设置超过可移动区域后，movable-view 是否还可以移动
x	number / string		定义 x 轴方向的偏移，x 的值如果不在可移动范围内，会自动移动到可移动范围内；改变 x 的值会触发动画
y	number / string		定义 y 轴方向的偏移，y 的值如果不在可移动范围内，会自动移动到可移动范围内；改变 y 的值会触发动画
damping	number	20	阻尼系数，用于控制 x 或 y 改变时的动画和过界回弹的动画，值越大，移动的速度越快
friction	number	2	摩擦系数，用于控制惯性滑动的动画，值越大，摩擦力越大，滑动的速度越快停止；必须大于 0，否则会被设置成默认值
disabled	boolean	false	设置是否禁用
scale	boolean	false	设置是否支持双指缩放，默认缩放手势生效区域在 movable-view 内

scale-min	number	0.5	设置定义缩放倍数最小值
scale-max	number	10	设置定义缩放倍数最大值
scale-value	number	1	设置定义缩放倍数，取值范围为 0.5～10
animation	boolean	true	设置是否使用动画
bindchange	eventhandle		拖动过程中触发的事件，event.detail = {x: x, y: y, source: source}，其中 source 表示产生移动的原因，值可为 touch（拖动）、touch-out-of-bounds（超出移动范围）、out-of-bounds（超出移动范围后的回弹）friction（惯性）和空字符串（setData）
bindscale	eventhandle		缩放过程中触发的事件，event.detail = {scale: scale}
htouchmove	eventhandle		初次手指触摸后移动为横向时触发，如果 catch 此事件，则意味着 touchmove 事件也被 catch
vtouchmove	eventhandle		初次手指触摸后移动为纵向时触发，如果 catch 此事件，则意味着 touchmove 事件也被 catch

movable-view 提供了 4 个事件：bindchange、bindscale、htouchmove 和 vtouchmove。

下面使用 movable-view 可移动视图容器组件来进行滑动，矩形区域代表可以移动的区域，其中的方块代表可以移动的组件，如图所示。



可移动视图容器

（1）在 WXML 文件里使用 movable-area 和 movable-view 视图容器组件进行布局，具体代码如下。

```
<view class="section">
  <movable-area style="height: 200px; width:100%; background: yellow;">
    <movable-view style="height: 50px; width: 50px; background: red;" x="{{x}}" y="{{y}}" direction="all" bindchange= "change" bindscale='scale' htouchmove="htouchmove" vtouchmove="vtouchmove">
    </movable-view>
  </movable-area>
</view>
```

（2）在 JS 文件里，提供拖动函数、缩放函数、初次手指触摸后移动为横向时触发函数、初次手指触摸后移动为纵向时触发函数，通过数据绑定的方式渲染到页面上，具体代码如下。

```
Page({
  data: {
    x: 0,
    y: 0
  },
  change: function (e) {
    console.log("拖动过程中触发的事件");
    console.log(e.detail)
  },
  scale: function (e) {
    console.log("缩放过程中触发的事件");
    console.log(e.detail)
  },
  htouchmove: function (e) {
    console.log("初次手指触摸后移动为横向时触发事件");
    console.log(e.detail)
  },
  vtouchmove: function (e) {
    console.log("初次手指触摸后移动为纵向时触发事件");
    console.log(e.detail)
  }
})
```

（3）当拖动可以移动的组件时，拖动过程中触发的事件就会触发，如拖动过程中，我们打印出拖动位置的日志，如图 4.11 所示。

cover-view 和 cover-image 是覆盖原生组件的视图容器组件。例如，在使用地图组件时，本身地图组件功能有局限，我们想放置一些特殊的内容或图片，就需要使用到覆盖地图组件的视图容器。

（1）cover-view 可以在原生组件之上覆盖文本视图，可覆盖的原生组件包括 map、video、canvas、camera，只支持嵌套 cover-view、cover-image。

（2）cover-image 可以在原生组件之上覆盖图片视图，可覆盖的原生组件与 cover-view 相同，支持嵌套在 cover-view 中。

下面使用 cover-view、cover-image 覆盖原生组件的视图组件，在 video 视频播放组件上放置播放、暂停两个图片，同时放置一个时间内容显示区域，如图所示。

视频播放　　覆盖视频播放组件

（1）在 WXML 文件里使用 cover-view、cover-image 覆盖原生组件的视图组件进行布局，具体代码如下。

```
<video id="myVideo" src="http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?filekey=30280201010421301f02016904025348041 02ca905ce620b1241b726bc41dcff44e00204012882540400&bizid=1023&hy=SH&fileparam=302c020101042530230204136ffd93020457e3c4ff02024ef202031e8d7f02030f424002040 45a320a0201000400" controls="{{false}}" event-model="bubble" style="width:100%">
  <cover-view class="controls">
    <cover-view class="play" bindtap="play">
      <cover-image class="img" src="../images/icon/play.jpg" />
    </cover-view>
    <cover-view class="pause" bindtap="pause">
      <cover-image class="img" src="../images/icon/pause.jpg" />
    </cover-view>
    <cover-view class="time">00:00</cover-view>
  </cover-view>
</video>
```

（2）在 WXSS 样式文件里添加样式，具体代码如下。

```
.controls {
  position: relative;
  top: 50%;
  height: 50px;
  margin-top: -25px;
  display: flex;
}
.play, .pause, .time {
  flex: 1;
  height: 100%;
}
.time {
  text-align: center;
  background-color: rgba(0, 0, 0, .5);
  color: white;
  line-height: 50px;
}
.img {
  width: 40px;
  height: 40px;
  margin: 5px auto;
}
```

（3）在 JS 文件里，提供视频播放、暂停函数，初始化视频播放组件，具体代码如下。

```
Page({
  onReady() {
    this.videoCtx = wx.createVideoContext('myVideo')
  },
  play() {
```

```
    this.videoCtx.play()
  },
  pause() {
    this.videoCtx.pause()
  }
})
```

4.2.6 项目实战：任务 9——实现搜索区域布局与海报轮播功能

1. 任务目标

通过实现搜索区域布局与海报轮播功能，学会利用 view 视图容器组件、image 图片组件来完成搜索区域布局，可以实现搜索区域布局水平居中和垂直居中；学会利用 swiper 滑块视图容器组件实现海报轮播功能。

莫凡商城首页顶部放置搜索区域和海报轮播区域，海报轮播区域可以动态地显示不同海报轮播内容，如图 4.14 和图 4.15 所示。

2. 任务实施

下面来实现首页搜索区域布局设计和海报轮播效果设计。

（1）在 index.wxml 文件里进行搜索区域布局设计，具体代码如下。

```
<view class="content">
  <view class="search">
    <view class="searchInput" bindtap="searchInput">
      <image src="/pages/images/tubiao/fangdajing-1.jpg" style="width:15px;height:19px;"></image>
      <text class="searchContent">搜索莫凡商品</text>
    </view>
  </view>
</view>
```

海报轮播效果 1

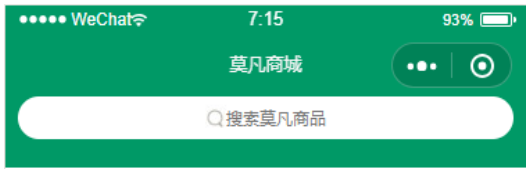
海报轮播效果 2

（2）在 index.wxss 文件里进行搜索区域布局样式渲染，具体代码如下。

```
.content{
  width: 100%;
  font-family: "Microsoft YaHei";
}
.search{
    width: 100%;
    background-color: #009966;
    height: 50px;
    line-height: 50px;
}
.searchInput{
```

```
    width: 95%;
    background-color: #ffffff;
    height: 30px;
    line-height: 30px;
    border-radius: 15px;
    display: flex;
    justify-content:center;
    align-items:center;
    margin: 0 auto;
}
.searchContent{
    font-size:12px;
    color: #777777;
}
```

（3）搜索区域界面布局的时候，使用了 view 组件、image 图片组件、text 组件，界面效果如图所示。



搜索区域布局设计

（4）在 index.wxml 文件里进行海报轮播效果布局设计，具体代码如下。

```
<view class="content">
  <view class="search">
    <view class="searchInput" bindtap="searchInput">
      <image src="/pages/images/tubiao/fangdajing-1.jpg" style="width:15px;height:19px;"></image>
      <text class="searchContent">搜索莫凡商品</text>
    </view>
  </view>
    <view class="haibao">
   <swiper indicator-dots="{{indicatorDots}}" autoplay="{{autoplay}}" interval="{{interval}}" duration="{{duration}}">
     <block wx:for="{{imgUrls}}">
       <swiper-item>
         <image src="{{item}}" class="silde-image" mode="scaleToFill"></image>
       </swiper-item>
     </block>
   </swiper>
  </view>
</view>
```

（5）在 index.wxss 文件里进行海报轮播效果样式渲染，具体代码如下。

```
.content{
  width: 100%;
  font-family: "Microsoft YaHei";
}
.search{
    width: 100%;
    background-color: #009966;
    height: 50px;
    line-height: 50px;
}
.searchInput{
    width: 95%;
    background-color: #ffffff;
    height: 30px;
    line-height: 30px;
    border-radius: 15px;
    display: flex;
    justify-content:center;
    align-items:center;
    margin: 0 auto;
}
.searchContent{
    font-size:12px;
    color: #777777;
}
```

```
.haibao{
    text-align: center;
    width: 100%;
}
.silde-image{
    width: 100%;
}
```

（6）在 index.js 文件里进行海报轮播效果数据初始化，具体代码如下。

```
Page({
  data: {
    indicatorDots: true,
    autoplay: true,
    interval: 5000,
    duration: 1000,
    imgUrls: [
      "/pages/images/haibao/1.jpg",
      "/pages/images/haibao/2.jpg",
      "/pages/images/haibao/3.jpg"
    ]
  }
})
```

这样就完成了海报轮播效果界面布局、界面渲染、页面数据初始化及绑定操作，海报轮播图可以动态地轮播显示不同的内容，如图所示。

海报轮播区域设计

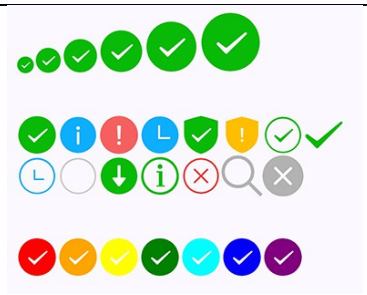
在设计海报轮播区域的时候，我们用到了 view 视图容器组件、swiper 滑块视图容器组件，利用 swiper 滑块视图容器组件来实现海报轮播效果，用到了 swiper 的 indicatorDots（是否显示指示点）、autoplay（是否自动轮播）、interval（间隔时长）、duration（滑动时长）属性。在渲染列表的时候，我们用到了 wx: for 列表渲染。swiper 滑块视图容器组件还有更多的属性，我们可以自己去尝试使用。

4.3 基础内容组件

使用小程序的基础内容组件，我们能快速进行各种页面的布局设计。基础内容组件包括 icon 图标组件、text 文本组件、progress 进度条组件、progress 进度条组件、editor 富文本编辑器等。

4.3.1 icon 图标组件

微信小程序提供了丰富的图标组件，应用于不同的场景，有成功、警告、提示、取消、下载等不同的含义，如图所示。



图标

icon 图标组件有 3 个属性：图标的类型 type、图标的大小 size 和图标的颜色 color，如表所示。

icon 的属性

属性	类型	默认值	说明
type	string		icon 的类型，有效值：success、success_no_circle、info、warn、waiting、cancel、download、search、clear
size	number	23	icon 的大小，单位为像素
color	color		icon 的颜色，与 CSS 的 color 相同

下面使用 icon 组件绘制出如图所示的图标。

（1）使用 icon 组件绘制不同尺寸的图标。

```
<view class="group">
    <icon type="success" size="20"/>
    <icon type="success" size="50"/>
    <icon type="success" size="60"/>
    <icon type="success" size="80"/>
    <icon type="success" size="100"/>
</view>
```

效果如图所示。

（2）使用 icon 组件绘制不同类型的图标。

```
<view class="group">
    <icon type="success" size="45"/>
    <icon type="info" size="45"/>
    <icon type="warn" size="45"/>
    <icon type="waiting" size="45"/>
    <icon type="safe_success" size="45"/>
    <icon type="success_circle" size="45"/>
    <icon type="success_no_circle" size="45"/>
    <icon type="waiting_circle" size="45"/>
    <icon type="circle" size="45"/>
    <icon type="download" size="45"/>
    <icon type="info_circle" size="45"/>
    <icon type="cancel" size="45"/>
    <icon type="search" size="45"/>
    <icon type="clear" size="45"/>
</view>
```

效果如图所示。

不同尺寸的图标

不同类型的图标

（3）使用 icon 组件绘制不同颜色的图标。

```
<view class="group">
  <icon type="success" size="45" color="red" />
  <icon type="success" size="45" color="orange" />
  <icon type="success" size="45" color="yellow" />
```

```
    <icon type="success" size="45" color="green" />
    <icon type="success" size="45" color="rgb(0, 255, 255)" />
    <icon type="success" size="45" color="blue" />
    <icon type="success" size="45" color="purple" />
  </view>
```

效果如图所示。

不同颜色的图标

这样就可以绘制出不同大小、不同类型、不同颜色的图标了，我们可以根据自己的需求，利用 icon 组件来设计小程序的图标。

4.3.2 text 文本组件

text 文本组件是用来放置文本信息的组件，它的属性如表所示。

text 的属性

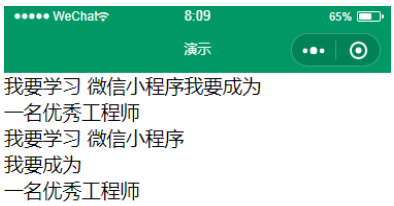
属性	类型	默认值	说明
selectable	boolean	false	文本是否可选
space	string	23	显示连续空格，ensp 表示中文字符空格一半大小，emsp 表示中文字符空格大小，nbsp 表示根据字体设置的空格大小
decode	boolean	false	是否解码

text 文本组件支持转义符“\”，如换行\n、空格\t，<text> 组件内只支持嵌套<text>组件，除了文本组件外的其他组件都无法长按选中。decode 属性可以解析的有不换行空格（ ）、小于号（<）、大于号（>）、&符号（&）、引号（'）、半角空格（ ）、全角空格（ ），各个操作系统的空格标准并不一致。

使用转义符的示例代码如下。

```
  <view class="btn-area">
    <view class="body-view">
      <text>我要学习\t 微信小程序</text>
      <text>我要成为\n 一名优秀工程师</text>
    </view>
    <view class="body-view">
      <text>我要学习\t 微信小程序</text>
    </view>
    <view class="body-view">
      <text>我要成为\n 一名优秀工程师</text>
    </view>
  </view>
```

界面效果如图所示。



转义符效果

从图中可以看出，\t 具有空格功能，\n 具有换行功能，同时也可以看出 text 文本组件是放置在一行里的，不同于 view 组件，每个 view 组件成一行。

4.3.3 progress 进度条组件

progress 进度条组件是一种用来提高用户体验度的组件，就像视频播放一样，可以通过进度条看到完整视频的长度、当前播放的进度，这样让用户能合理地安排自己的时间，提升用户体验，微信小程序也提供了 progress 进度条组件，它的属性如表所示。

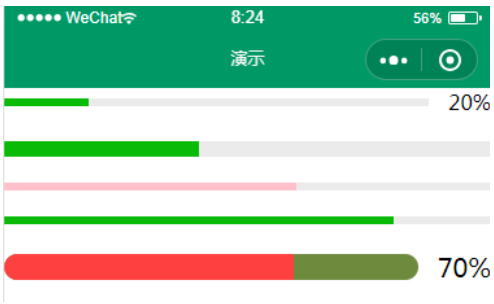
进度条属性

属性	类型	默认值	说明
percent	number	无	百分比，值为 0～100
show-info	boolean	false	在进度条右侧显示百分比
border-radius	number/string	0	圆角大小
font-size	number/string	16	右侧百分比字体大小
stroke-width	number	6	进度条线的宽度，单位为像素
color	string	#09BB07	进度条的颜色
activeColor	string	#09BB07	已选择的进度条的颜色
backgroundColor	string	#EBEBEB	未选择的进度条的颜色
active	boolean	false	进度条从左往右的动画
active-mode	string	backwards	backwards 表示动画从头播；forwards 表示动画从上次结束点接着播
bindactiveend	eventhandle		动画完成事件

可以尝试各种进度条的效果，示例代码如下。

```
<progress percent="20" show-info />
<progress percent="40" stroke-width="12" />
<progress percent="60" color="pink" />
<progress percent="80" active />
<progress percent="70" show-info stroke-width="20" border-radius="10" font-size="20" color="#CCCCCC" activeColor="#FF4040" backgroundColor="#6E8B3D" active active-mode="backwards" />
```

界面效果如图所示。



进度条效果

4.3.4 rich-text 富文本组件

通过 rich-text 富文本组件可以在 WXML 页面文件中显示一些富文本内容，如显示 HTML 的一些元素内容。rich-text 的属性如表所示。

rich-text 的属性

属性	类型	默认值	说明
nodes	array/string	[]	节点列表/HTML
space	string		显示连续空格，ensp 为中文字符空格一半大小、emsp 为中文字符空格大小，nbsp 为根据字体设置的空格大小

rich-text 的 nodes 节点列表属性推荐使用 Array 类型。nodes 支持两种节点，通过 type 来区分，分别是元素节点和文本节点，默认为元素节点，即在 rich-text 富文本区域里显示的 HTML 节点。

1. 元素节点（type= node）

元素节点的属性如表所示，受信任的 HTML 节点包括 a、abbr、address、article、aside、b、bdi、bdo、dir、big、blockquote、br、caption、center、cite、code、col、colgroup、dd、del、div、dl、dt、em、fieldset、font、footer、h1、h2、h3、h4、h5、h6、header、hr、i、img、ins、label、legend、li、mark、nav、ol、p、pre、q、rt、ruby、s、section、small、span、strong、sub、sup、table、tbody、td、tfoot、th、thead、tr、tt、u、ul。

元素节点的属性

属性	类型	默认值	说明
name	标签名	string	支持部分受信任的 HTML 节点
attrs	属性	Object	支持部分受信任的属性，遵循 Pascal 命名法
children	子节点列表	Array	结构和 nodes 一致

2. 文本节点（type= text）

文本节点的属性如表所示。

文本节点的属性

属性	类型	默认值	说明
text	文本	string	支持 entities

示例代码如下。

```
<rich-text nodes="{{nodes}}" bindtap="tap"></rich-text>
```

```
Page({
  data: {
    nodes: [{
      name: 'div',
      attrs: {
        class: 'div_class',
        style: 'line-height: 60px; color: red;'
      },
      children: [{
        type: 'text',
        text: 'Hello World!'
      }]
    }]
  },
  tap() {
    console.log('tap')
  }
})
```

注意：

（1）nodes 不推荐使用 string 类型，如果使用 string 类型，组件会将 string 类型转换为 Array 类型，导致性能有所下降。

（2）rich-text 组件内屏蔽所有节点的事件。

（3）attrs 属性不支持 id，支持 class。

（4）name 属性对大小写不敏感。

（5）如果使用了不受信任的 HTML 节点，该节点及其所有子节点将会被移除。

（6）img 标签仅支持网络图片。

（7）如果在自定义组件中使用 rich-text 组件，那么仅自定义组件的 WXSS 样式对 rich-text 中的 class 生效。

4.3.5 editor 富文本编辑器及 API

editor 富文本编辑器，可以对图片、文字进行编辑，可以导出带标签的 html 和纯文本的 text 内容，富文本组件内部引入了一些基本的样式使得内容可以正确展示，开发时可以进行覆盖。需要注意的是，在其他组件或环境中使用富文本组件导出的 html 时，需要额外引入这段样式，并维护 <ql-container><ql-editor></ql-editor></ql-container>的结构，editor 的属性如表所示。

editor 的属性

属性	类型	默认值	说明
read-only	boolean	false	设置编辑器为只读
placeholder	string		提示信息
show-img-size	boolean	false	单击图片时显示图片大小控件
show-img-toolbar	boolean	false	单击图片时显示工具栏控件
show-img-resize	boolean	false	单击图片时显示修改尺寸控件
bindready	eventhandle		编辑器初始化完成时触发
bindfocus	eventhandle		编辑器聚焦时触发，event.detail = {html, text, delta}

bindblur	eventhandle		编辑器失去焦点时触发，detail = {html, text, delta}
bindinput	eventhandle		编辑器内容改变时触发，detail = {html, text, delta}
bindstatuschange	eventhandle		通过 Context 方法改变编辑器内样式时触发，返回选区已设置的样式

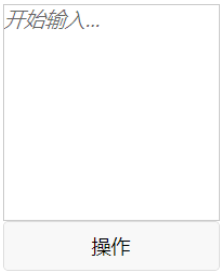
示例代码如下。

```
<view class="container">
  <view class="page-body">
    <editor id="editor" class="ql-container" placeholder="{{placeholder}}" bindready="onEditorReady" read-only= "{{readOnly}}" bindinput="onContentChange" style="border:1px solid #cccccc;width:200px;" showImgSize showImgToolbar showImgResize>
    </editor>
    <view>
      <button bindtap="clickBtn">操作</button>
    </view>
  </view>
</view>
```

```
Page({
  data: {
    placeholder: '开始输入...',
    isReadOnly: false
  },
  onEditorReady:function() {//初始化编辑器
    var that = this;
    wx.createSelectorQuery().select('#editor').context(function (res) {
      that.editorCtx = res.context;
    }).exec()
  },
  onContentChange:function(e){//监控编辑器内容变化
    console.log(e.detail);
  },
  clickBtn:function(e) {//操作
    //清空编辑器内容
    this.editorCtx.clear();
    //插入文本内容
    this.editorCtx.insertText({
      text: "插入内容"
    });
    //插入图片
    this.editorCtx.insertImage({
      src: "https://api.mofun365.com:8888/images/banner/1555848473813.jpg"
    });
    //初始化编辑器内容
    this.editorCtx.setContents({
      html: "<h1>初始化编辑器内容<h1>"
    });
    //获取编辑器内容
    this.editorCtx.getContents({
      success:function(res) {
        console.log(res);
      }
    });
    //修改样式
    this.editorCtx.format("align", "center");
    //清除当前选区的样式
    this.editorCtx.removeFormat({
      success: function (res) {
        console.log("------------------清除当前选区的样式------------------");
      }
    });
    //插入分割线
    this.editorCtx.insertDivider({
      success: function (res) {
        console.log("------------------插入分割线------------------");
      }
    });
    //恢复
```

```
      this.editorCtx.redo({
        success: function (res) {
          console.log("------------------恢复-----------------");
        }
      });
      //撤销
      this.editorCtx.undo({
        success: function (res) {
          console.log("------------------撤销-----------------");
        }
      });
    }
  })
```

编辑器效果如图所示。



编辑器效果

（1）在 editor 组件上定义 id 属性，然后在 onEditorReady 函数里初始化富文本编辑器，获取 EditorContext 编辑器上下文对象，操作 editor 组件需要先将 EditorContext 实例化，具体代码如下。

```
  onEditorReady:function() {//初始化编辑器
    var that = this;
    wx.createSelectorQuery().select('#editor').context(function (res) {
      that.editorCtx = res.context;
    }).exec()
  }
```

（2）用 EditorContext.clear()清空编辑器内容。

```
  this.editorCtx.clear();
```

（3）用 EditorContext.insertText（Object object）插入文本内容，是覆盖当前选区内容，重新设置一段文本内容。

```
  this.editorCtx.insertText({
    text: "插入内容"
  })
```

（4）用 EditorContext.insertImage（Object object）插入图片，提供图片地址，仅支持 http（或 https）和 base64 格式。

```
  this.editorCtx.insertImage({
    src: "https://api.mofun365.com:8888/images/banner/1555848473813.jpg"
  });
```

（5）用 EditorContext. getContents() 获取编辑器内容。

```
  this.editorCtx.getContents({
    success:function(res) {
      console.log(res);
    }
  });
```

（6）用 EditorContext.format（string name, string value）修改样式。

```
  this.editorCtx.format("align", "center");
```

（7）用 EditorContext.removeFormat()清除当前选区的样式。

```
  this.editorCtx.removeFormat({
    success: function (res) {
      console.log("------------------清除当前选区的样式-----------------");
    }
  });
```

（8）用 EditorContext.insertDivider()插入分割线。

```
  this.editorCtx.insertDivider({
    success: function (res) {
      console.log("------------------插入分割线-----------------");
```

```
        }
      });
```

（9）用 EditorContext.redo()恢复之前的操作。

```
  this.editorCtx.redo({
      success: function (res) {
        console.log("------------------恢复-----------------");
      }
  });
```

（10）用 EditorContext.undo()撤销之前操作。

```
  this.editorCtx.undo({
      success: function (res) {
        console.log("------------------撤销-----------------");
      }
    });
```

小结	本节课讲解了视图容器组件，包括 view 视图容器组件、scroll-view 可滚动视图容器组件、swiper 滑块视图容器组件、movable-view 可移动视图容器组件、cover-view 覆盖原生组件的视图容器组件；讲解了基础内容组件，包括 icon 图标组件、text 文本组件、progress 进度条组件、rich-text 富文本组件、editor 富文本编辑器；通过这些组件的使用来完成搜索区域布局与海报轮播功能。
练习	