第5章 视图容器组件

通过前面内容的学习，读者对小程序有了一个大致的了解，包括组成小程序的各个文件，它们的作用是什么，以及小程序的 MINA 框架是怎么回事儿。那么从这一章开始，就要开始学习小程序开发的具体细节。

本章的主要内容是对视图容器组件的介绍。视图容器组件，简单来说与 HTML 中的 < div > 标签一样，是小程序页面布局的基础元素，用来组织元素的排布，设置页面的整体布局。当然小程序的视图容器组件要比单纯的< div > 丰富得多，掌握了这部分内容，才能更好地设计出页面更加合理美观的小程序。

本章学习目标：

- 了解 Flex 布局的方式。
- 掌握对 view 组件、scroll-view 组件、swiper 组件、movable-view 组件、movable-area 组件、cover-view 组件、cover-image 组件的使用。

5.1 Flex 布局和 view 组件

【任务要求】

新建一个如图 5-1 所示的小程序页面，了解小程序页面组件的基本排列方式。

【任务分析】

本任务主要练习的是对 view 组件和微信小程序所采用的 Flex 布局的操作。观察图 5-1 可以看到，整个页面的元素整体上来看是纵向排列的，而其中又插入了一个横向布局的三个方块和一个纵向布局的三个方块。因此需要对最外层设置成纵向排列，同时单独设置横向布局的三个方块为横向排列。

【任务操作】

(1) 打开示例项目，并在其 app.json 文件中新注册一个页面"pages/Chapter_5/5_1_view/5_1_view"。同时修改窗口的配置，使其达到如图 5-1 所示的效果。修改完成后的 app.json 文件如下。

```
{
  "pages": [
    "pages/Chapter_3/mina/mina",
    "pages/Chapter_3/page/page",
    "pages/Chapter_3/WXML/WXML",
    "pages/Chapter_3/WXKEY/WXKEY",
    "pages/Chapter_3/WXSS/WXSS",
```

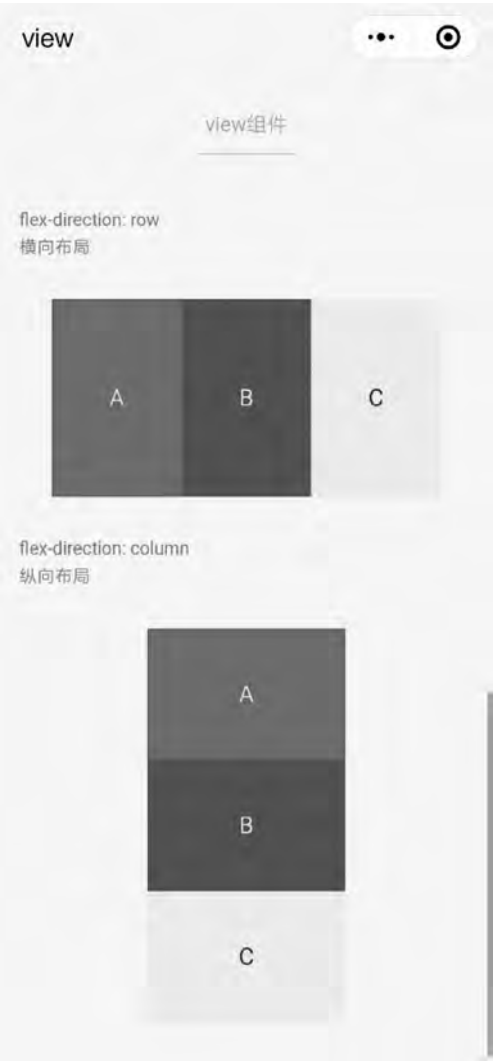

图 5-1　view 组件布局

```
    "pages/Chapter_4/frontend/frontend",
    "pages/index/index",
    "pages/logs/logs",
    "pages/Chapter_5/5_1_view/5_1_view"
  ],
  "window": {
    "navigationBarTextStyle": "black",
    "navigationBarTitleText": "演示",
    "navigationBarBackgroundColor": "#F8F8F8",
    "backgroundColor": "#F8F8F8"
  }
}
```

保存文件并编译项目,让开发者工具自动生成所需的目录和页面。

(2) 修改 5_1_view.json 文件为如下所示代码,让窗口显示“view”。

```
{
  "navigationBarTitleText": "view"
}
```

(3) 在 5_1_view.wxml 文件中写入如下代码,填充页面元素。涉及本任务关键的元素会加粗表示。

```
<!-- pages/Chapter_5/5_1_view/5_1_view.wxml -->
<view class="container">
  <view class="page-head">
    <view class="page-head-title">view 组件</view>
    <view class="page-head-line"></view>
  </view>
  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title">
        <text>flex-direction: row\n 横向布局</text>
      </view>
      <view class="page-section-spacing">
        <view class="flex-wrp" style="flex-direction:row;">
          <view class="flex-item demo-text-1"></view>
          <view class="flex-item demo-text-2"></view>
          <view class="flex-item demo-text-3"></view>
        </view>
      </view>
    </view>
    <view class="page-section">
      <view class="page-section-title">
        <text>flex-direction: column\n 纵向布局</text>
      </view>
      <view class="flex-wrp" style="flex-direction:column;">
        <view class="flex-item flex-item-V demo-text-1"></view>
        <view class="flex-item flex-item-V demo-text-2"></view>
        <view class="flex-item flex-item-V demo-text-3"></view>
      </view>
    </view>
  </view>
</view>
```

(4) 在 5_1_view.wxss 文件中编写如下代码,实现对页面的样式调整。涉及本任务重点的样式设置将会用粗体显示。

```
/* pages/Chapter_5/5_1_view/5_1_view.wxss */
page {
  background-color: #F8F8F8;
  height: 100%;
  font-size: 32rpx;
  line-height: 1.6;
}
```

```
.container {
  display: flex;
  flex-direction: column;
  min-height: 100%;
  justify-content: space-between;
  font-size: 32rpx;
  font-family: -apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
.page-head{
  padding: 60rpx 50rpx 80rpx;
  text-align: center;
}
.page-head-title{
  display: inline-block;
  padding: 0 40rpx 20rpx 40rpx;
  font-size: 32rpx;
  color: #BEBEBE;
}
.page-head-line{
  margin: 0 auto;
  width: 150rpx;
  height: 2rpx;
  background-color: #D8D8D8;
}
.page-body {
  width: 100%;
  flex-grow: 1;
  overflow-x: hidden;
}
.page-section{
  width: 100%;
  margin-bottom: 60rpx;
}
.page-section-title{
  font-size: 28rpx;
  color: #999999;
  margin-bottom: 10rpx;
  padding-left: 30rpx;
  padding-right: 30rpx;
}
.page-section-spacing{
  box-sizing: border-box;
  padding: 0 80rpx;
}
.flex-wrp{
  margin-top: 60rpx;
  display:flex;
}
.flex-item{
  width: 200rpx;
  height: 300rpx;
```

```
  font-size: 26rpx;
}
.demo-text-1{
  position: relative;
  align-items: center;
  justify-content: center;
  background-color: #1AAD19;
  color: #FFFFFF;
  font-size: 36rpx;
}
.demo-text-1:before{
  content: 'A';
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
.demo-text-2{
  position: relative;
  align-items: center;
  justify-content: center;
  background-color: #2782D7;
  color: #FFFFFF;
  font-size: 36rpx;
}
.demo-text-2:before{
  content: 'B';
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
.demo-text-3{
  position: relative;
  align-items: center;
  justify-content: center;
  background-color: #F1F1F1;
  color: #353535;
  font-size: 36rpx;
}
.demo-text-3:before{
  content: 'C';
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

```
.flex-item-V{
  margin: 0 auto;
  width: 300rpx;
  height: 200rpx;
}
```

(5) 在前面的任务中,均是将需要预览的页面放在 app.json 文件里 pages 数组的第一项,这样固然方便,但是当需要调试多个页面时,不停去修改 app.json 文件也比较麻烦。因此从这一部分起,将使用设置"编译模式"的方式来对页面进行预览。在工具栏中打开"普通编译"下拉框,选择"添加编译模式"(如图 5-2 所示),在弹出的对话框中,给"模式名称"起名为"view",从"启动页面"下拉框中选择"pages/Chapter_5/5_1_view/5_1_view"为启动页面(如图 5-3 所示)。

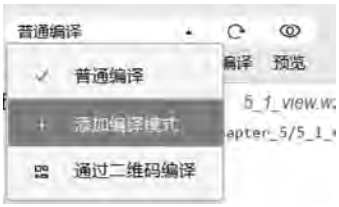

图 5-2 添加编译模式

图 5-3 自定义编译条件将页面 5_1_view 设置为启动页面

(6) 如图 5-4 所示,选择刚刚新建的 view 编译模式,单击"编译"按钮,便可以在界面左边的模拟器中查看到刚刚编写的 5_1_view 页面的效果了。

【相关知识】

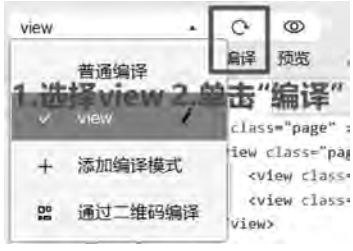

图 5-4 使用自定义编译模式编译

要想学习小程序的前端页面设计,Flex 布局是一个非常重要的部分。和传统的布局解决方案使用"盒状模型",依赖 display+position+float 属性来控制元素的排列位置不同的是,于 2009 年由 W3C(World Wide Web Consortium,万维网联盟)提出的新的 Flex 布局方案,可以简便、完整、响应式地实现各种页面布局。目前,它已经得到了所有浏览器的支持。在默认情况下,Flex 布局是从左向右水平依次放置组件,或者是从上到下垂直依次放置组件。当一个< view ></view >标签的样式属性 display 的值设为 flex 时,便表示使用了 Flex 弹性布局方案。在本任务的实现过程中,使用到的 flex 样式属性见表 5-1。

表 5-1　常见 Flex 样式属性说明

属　性	作　用	可　选　值	说　明
flex-direction	表示元素的排列方式	row	元素横向排列
		column	元素纵向排列
justify-content	表示元素在主轴上的排列方式。如果元素为横向排列，则主轴为水平轴	flex-start	紧挨着主轴开始处对齐
		flex-end	紧挨着主轴结尾处对齐
		center	在主轴居中处对齐
		space-between	元素平均分布在主轴上
		space-around	元素平均分布在主轴上，两边留有一半的间隔空间
align-items	表示元素在侧轴上的排列方式。如果元素为横向排列，则侧轴为纵轴	stretch	默认值，元素被拉伸以适应容器
		center	元素位于侧轴中心
		flex-start	元素在侧轴开始处
		flex-end	元素在侧轴结尾处
		baseline	元素位于容器内基线上

在上面任务的实现过程中，可以看到，在 5_1_view.wxss 文件中，container 类的 display 属性值为 Flex，表示整个页面布局采取的是 flex 方案；flex-direction 属性值为 column，表示元素整体为纵向排列；justify-content 属性值为 space-between，表示元素平均分布在主轴（也就是纵轴）上。同时，在 demo-text-1、demo-text-2 和 demo-text-3 这三个类中，align-items 和 justify-content 这两个属性值都被设为 center，使得色块中的文本 A、B、C 能在色块中水平、垂直都居中显示。

在文件 5_1_view.wxml 中可以看到，第一个横向排列的色块组合，在< view ></view >标签中使用了内联样式 style="flex-direction:row;"，表示里面包含的三个色块采用横向排列，第二个纵向排列的三个色块组合，在< view ></view >标签中使用了内联样式 style="flex-direction:column;"，表示里面包含的三个色块采用的是纵向排列。因此，通过对元素的 flex 相关的属性进行设置，得到了任务要求所展示的元素排列。

除了 Flex 样式的相关属性设置和在 3.6 节组件部分提到的所有组件共有的属性外，view 组件还包含的属性见表 5-2。

表 5-2　view 组件属性

属　性　名	类型	默认值	说　明	最低版本
hover-class	String	none	指定按下去的样式类。当 hover-class="none" 时，没有效果	
hover-stop-propagation	Boolean	false	指定是否阻止本节点的祖先节点出现单击态	1.5.0
hover-start-time	Number	50	按住后多久出现单击态，单位：毫秒	
hover-stay-time	Number	400	手指松开后单击态保留时间，单位：毫秒	

表 5-2 中的属性主要是给用户的单击操作提供视觉反馈，例如，如果要让一个 view 组件（以任务中的 A 色块为例）在被按住时背景颜色透明度发生改变，可以在 view 组件中编

写如下代码。

```
<view class="flex-item demo-text-1" hover-class='change-color'></view>
```


然后定义类 change-color 的样式为：

```
.change-color{
  background: rgba(26, 173, 25, 0.7);
}
```

因此，每当该 view 组件（A 色块）被单击时，其背景透明度就将变成 0.7，可以给用户一个直观的视觉反馈效果。

5.2 滚动视图组件 scroll-view

【任务要求】

使用滚动视图组件 scroll-view，使 A、B、C 三个色块能如图 5-5 所示纵向滚动和横向滚动。同时监听滚动、滚动到顶部、滚动到底部的事件，在控制台观察事件输出。

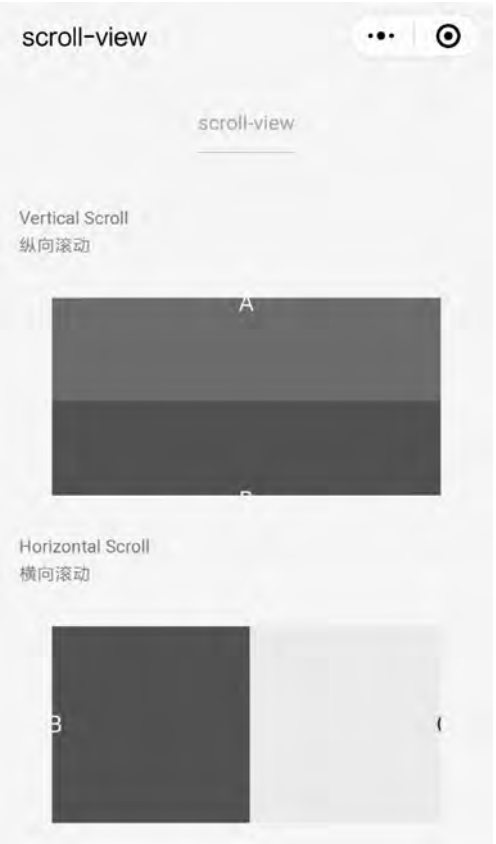

图 5-5 scroll-view 组件任务示例

【任务分析】

本任务主要是练习 scroll-view 组件的使用。从图 5-5 可以看出，主要包含纵向滚动和横向滚动两部分。要想观察相关滚动事件的输出，还需要绑定滚动事件 bindscroll，滚动到顶部/左边事件 bindscrolltoupper，滚动到底部/右边事件 bindscrolltolower。针对这些事件，需要在 js 文件中编写对应的处理函数，将事件详情输出在控制台中。

【任务操作】

(1) 打开示例项目，在 app.json 文件的 pages 数组中新增一项"pages/Chapter_5/5_2_scroll-view/5_2_scroll-view"。保存并编译项目，让开发者工具自动生成必要的目录和页面文件。

(2) 修改 5_2_scroll-view.json 中的内容为如下代码，使页面窗口标题显示为 scroll-view。

```
{
  "navigationBarTitleText": "scroll-view"
}
```

(3) 在 5_2_scroll-view.wxml 文件中编写如下代码，排列好页面元素。

```
<!--pages/Chapter_5/5_2_scroll-view/5_2_scroll-view.wxml-->
<view class="container">
  <view class="page-head">
    <view class="page-head-title">scroll-view</view>
    <view class="page-head-line"></view>
  </view>

  <view class="page-body">
    <view class="page-section">
      <view class="page-section-title">
        <text>Vertical Scroll\n纵向滚动</text>
      </view>
      <view class="page-section-spacing">
        <scroll-view scroll-y="true" style="height: 300rpx;" bindscrolltoupper="upper" bindscrolltolower="lower" bindscroll="scroll">
          <view class="scroll-view-item demo-text-1"></view>
          <view class="scroll-view-item demo-text-2"></view>
          <view class="scroll-view-item demo-text-3"></view>
        </scroll-view>
      </view>
    </view>
    <view class="page-section">
      <view class="page-section-title">
        <text>Horizontal Scroll\n横向滚动</text>
      </view>
      <view class="page-section-spacing">
        <scroll-view class="scroll-view_H" scroll-x="true" bindscroll="scroll" style="width: 100%">
          <view class="scroll-view-item_H demo-text-1"></view>
          <view class="scroll-view-item_H demo-text-2"></view>
```

```
            <view class="scroll-view-item_H demo-text-3"></view>
          </scroll-view>
        </view>
      </view>
    </view>
</view>
```

(4) 在 5_2_scroll-view.js 文件中,添加对应的 scroll 函数,upper 函数和 lower 函数用来处理滚动事件,滚动到顶部事件和滚动到底部事件。完成后的 5_2_scroll-view.js 文件内容如下。和本任务相关的函数已加粗显示。

```
// pages/Chapter_5/5_2_scroll-view/5_2_scroll-view.js
Page({
  data: {},
  onLoad: function (options) {},
  onReady: function () {},
  onShow: function () {},
  onHide: function () {},
  onUnload: function () {},
  onPullDownRefresh: function () {},
  onReachBottom: function () {},
  onShareAppMessage: function () {},
  upper(e) {
    console.log(e)
  },
  lower(e) {
    console.log(e)
  },
  scroll(e) {
    console.log(e)
  }
})
```

(5) 在 5_2_view.wxss 文件中写入如下内容,完成对页面样式的调整。

```
/* pages/Chapter_5/5_2_scroll-view/5_2_scroll-view.wxss */
page {
  background-color: #F8F8F8;
  height: 100%;
  font-size: 32rpx;
  line-height: 1.6;
}
.container {
  display: flex;
  flex-direction: column;
  min-height: 100%;
  justify-content: space-between;
  font-size: 32rpx;
  font-family: -apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
.page-head{
```

```
  padding: 60rpx 50rpx 80rpx;
  text-align: center;
}
.page-head-title{
  display: inline-block;
  padding: 0 40rpx 20rpx 40rpx;
  font-size: 32rpx;
  color: #BEBEBE;
}
.page-head-line{
  margin: 0 auto;
  width: 150rpx;
  height: 2rpx;
  background-color: #D8D8D8;
}
.page-body {
  width: 100%;
  flex-grow: 1;
  overflow-x: hidden;
}
.page-section{
  width: 100%;
  margin-bottom: 60rpx;
}
.page-section-title{
  font-size: 28rpx;
  color: #999999;
  margin-bottom: 10rpx;
  padding-left: 30rpx;
  padding-right: 30rpx;
}
.page-section-spacing{
  margin-top: 60rpx;
  box-sizing: border-box;
  padding: 0 80rpx;
}
.scroll-view_H{
  white-space: nowrap;
}
.scroll-view-item{
  height: 300rpx;
}
.scroll-view-item_H{
  display: inline-block;
  width: 100%;
  height: 300rpx;
}
.demo-text-1{
  position: relative;
  align-items: center;
  justify-content: center;
```

```
    background-color: #1AAD19;
    color: #FFFFFF;
    font-size: 36rpx;
  }
  .demo-text-1:before{
    content: 'A';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .demo-text-2{
    position: relative;
    align-items: center;
    justify-content: center;
    background-color: #2782D7;
    color: #FFFFFF;
    font-size: 36rpx;
  }
  .demo-text-2:before{
    content: 'B';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .demo-text-3{
    position: relative;
    align-items: center;
    justify-content: center;
    background-color: #F1F1F1;
    color: #353535;
    font-size: 36rpx;
  }
  .demo-text-3:before{
    content: 'C';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
```

(6) 添加一个名叫 scroll-view 的编译模式，将页面 5_2_scroll-view 设置为启动页，单击“编译”按钮，就可以在左边的模拟器中看到新建的页面的效果了。在模拟器中的纵向滚动区域滚动鼠标，可以在下方控制台看到对应事件的输出结果，分别如图 5-6～图 5-8 所示。

【相关知识】

scroll-view 表示的是可滚动的视图区域。其包含的属性见表 5-3。

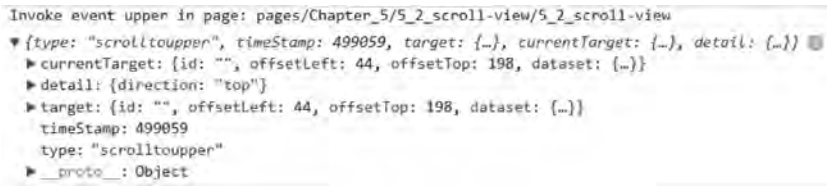

图 5-6　输出滚动到顶部事件详情

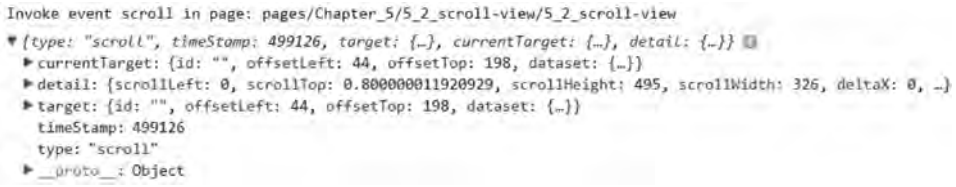

图 5-7　输出滚动事件详情

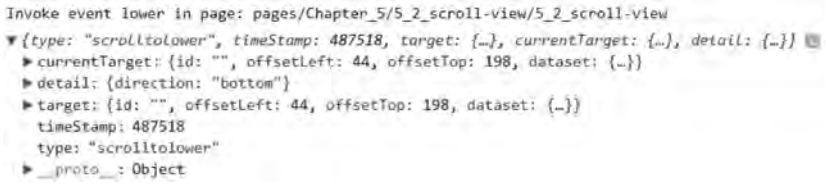

图 5-8　输出滚动到底部事件详情

表 5-3　scroll-view 组件属性说明

属　性　名	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 / String	50	距顶部/左边多远时(单位：px，2.4.0 起支持 rpx)，触发 scrolltoupper 事件
lower-threshold	Number / String	50	距底部/右边多远时(单位：px，2.4.0 起支持 rpx)，触发 scrolltolower 事件
scroll-top	Number / String		设置竖向滚动条位置(单位：px，2.4.0 起支持 rpx)
scroll-left	Number / String		设置横向滚动条位置(单位：px，2.4.0 起支持 rpx)
scroll-into-view	String		值应为某子元素 id(id 不能以数字开头)。设置哪个方向可滚动，则在哪个方向滚动到该元素
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，只支持竖向
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}

在本次任务中，第一组三个色块，通过设置< scroll-view ></scroll-view >标签的 scroll-y 属性值为 true，实现了色块的纵向滚动。需要注意的是，使用竖向滚动时，需要通过 WXSS 设置 height，给< scroll-view >一个固定高度。在本任务的第一个纵向滚动的 scroll-view 中，设置了内联样式 style="height: 300rpx;"，固定了整个纵向滚动区域的高度为 300rpx。在第二个滚动区域中，设置了< scroll-view ></scroll-view >标签的 scroll-x 属性值为 true，使得三个色块可以横向滚动。

在第一个纵向滚动的 scroll-view 组件中，还对 scrolltoupper 事件通过设置 bindscrolltoupper 属性的值绑定了函数 upper( )，对 scrolltolower 事件通过设置 bindscrolltolower 属性的值绑定了函数 lower()，对 scroll 事件通过设置 bindscroll 属性的值绑定了函数 scroll()。在第二个横向滚动的 scroll-view 组件中，同样绑定了 scroll 事件，并通过函数 scroll()来处理。在 5_2_scroll-view.js 文件中，upper()函数、lower()函数和 scroll()函数均是将事件详情进行了直接的输出。在输出的内容中，除了事件均包含的公共内容外，其中的 event.detail 包含当前元素的一些位置信息。scrollLeft 表示该元素显示(可见)的内容与该元素实际内容左边的距离，因此该值在第一个纵向滚动区域触发的 scroll 事件中为零，在第二个横向滚动区域触发的 scroll 事件中会随着元素的左右滚动发生变化。scrollTop 表示该元素显示(可见)的内容与该元素实际内容上边的距离，因此该值在第一个纵向滚动区域触发的 scroll 事件中会随着元素的上下滚动而变化，在第二个横向滚动区域触发的 scroll 事件中为零。scrollHeight 表示元素的总高度，scrollWidth 表示元素的总宽度，均包括由于溢出而无法展示在网页的不可见部分。deltaX 和 deltaY 则分别表示在横向上和纵向上元素移动的距离，纵向滚动的话，deltaX 的值为零，横向滚动的话，deltaY 的值为零。

使用 scroll-view 组件除了前面提到的纵向滚动需要设置组件的固定高度外，还有以下几点需要注意的地方。

(1) 请勿在 scroll-view 中使用 textarea、map、canvas、video 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 onPullDownRefresh 的；

(4) 若要使用下拉刷新，请使用页面的滚动，而不是 scroll-view，这样也能通过单击顶部状态栏回到页面顶部。

5.3 滑块视图容器 swiper

【任务要求】

创建一个页面，如图 5-9 所示排列 A、B、C 三个色块，通过使用 swiper 组件使其可以横向滑动。同时增加一个按钮，动态控制是否显示指示点。

【任务分析】

本次任务主要是练习 swiper 组件的使用。除了基本的排列显示三个色块外，还增加了一个动态控制指示点显示的功能，可以通过监听按钮的单击事件，动态修改 swiper 的 indicator-dots 属性值来实现。

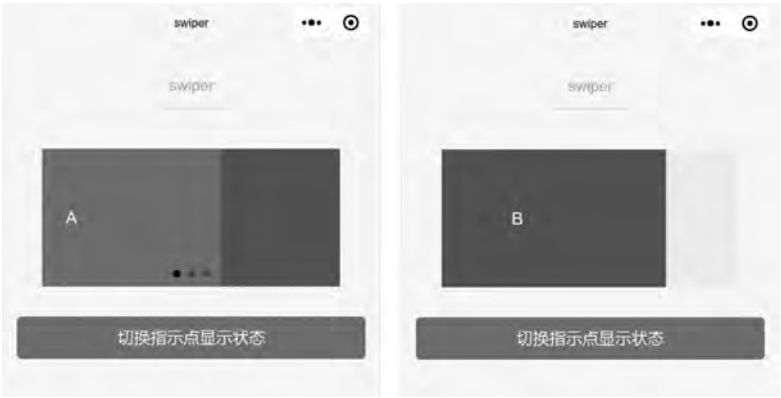

图 5-9　swiper 组件有指示点(左)和无指示点(右)示例

【任务操作】

(1) 打开示例小程序项目,在 app.json 文件的 pages 数组中新增页面“pages/Chapter_5/5_3_swiper/5_3_swiper”,单击“编译”按钮,生成 5_3_swiper 页面所需的文件。

(2) 打开 5_3_swiper.json 文件,修改其中内容为如下代码,使页面窗口标题栏显示“swiper”。

```
{
  "navigationBarTitleText": "swiper"
}
```

(3) 打开 5_3_swiper.wxml 文件,修改其中内容为如下代码,确定页面结构。

```
<!-- pages/Chapter_5/5_3_swiper/5_3_swiper.wxml -->
<view class="container">
  <view class="page-head">
    <view class="page-head-title">swiper</view>
    <view class="page-head-line"></view>
  </view>

  <view class="page-body">
    <view class="page-section page-section-spacing swiper">
      <swiper indicator-dots="{{indicatorDots}}">
        <swiper-item>
          <view class="swiper-item demo-text-1"></view>
        </swiper-item>
        <swiper-item>
          <view class="swiper-item demo-text-2"></view>
        </swiper-item>
        <swiper-item>
          <view class="swiper-item demo-text-3"></view>
        </swiper-item>
      </swiper>
    </view>

    <view class="page-section">
```

```
      <view class="btn-area">
        <button bindtap="changeIndicatorDots" type="primary">切换指示点显示状态
</button>
      </view>
    </view>
  </view>
</view>
```

(4) 在 5_3_swiper.js 文件中,新增 changeIndicatorDots()函数,用来响应按钮的单击事件,修改指示点的显示状态。同时在 data 数组中,初始化 indicatorDots 的值为 true,即默认显示指示点。代码如下。

```
// pages/Chapter_5/5_3_swiper/5_3_swiper.js
Page({
  data: {
    indicatorDots: true
  },
  onLoad: function (options) {},
  onReady: function () {},
  onShow: function () {},
  onHide: function () {},
  onUnload: function () {},
  onPullDownRefresh: function () {},
  onReachBottom: function () {},
  onShareAppMessage: function () {},
  changeIndicatorDots() {
    this.setData({
      indicatorDots: !this.data.indicatorDots
    })
  }
})
```

(5) 打开 5_3_swiper.wxss 文件,写入如下代码,完成对页面样式的设置。

```
/* pages/Chapter_5/5_3_swiper/5_3_swiper.wxss */
page {
  background-color: #F8F8F8;
  height: 100%;
  font-size: 32rpx;
  line-height: 1.6;
}
.container {
  display: flex;
  flex-direction: column;
  min-height: 100%;
  justify-content: space-between;
  font-size: 32rpx;
  font-family: -apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
.page-head{
  padding: 60rpx 50rpx 80rpx;
```

```
    text-align: center;
}
.page-head-title{
    display: inline-block;
    padding: 0 40rpx 20rpx 40rpx;
    font-size: 32rpx;
    color: #BEBEBE;
}
.page-head-line{
    margin: 0 auto;
    width: 150rpx;
    height: 2rpx;
    background-color: #D8D8D8;
}
.page-body {
    width: 100%;
    flex-grow: 1;
    overflow-x: hidden;
}
.page-section{
    width: 100%;
    margin-bottom: 60rpx;
}
.page-section-title{
    font-size: 28rpx;
    color: #999999;
    margin-bottom: 10rpx;
    padding-left: 30rpx;
    padding-right: 30rpx;
    padding: 0;
}
.page-section-spacing{
    box-sizing: border-box;
    padding: 0 80rpx;
}
.swiper-item{
    display: block;
    height: 150px;
}
.demo-text-1{
    position: relative;
    align-items: center;
    justify-content: center;
    background-color: #1AAD19;
    color: #FFFFFF;
    font-size: 36rpx;
}
.demo-text-1:before{
    content: 'A';
    position: absolute;
    top: 50%;
```

```
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .demo-text-2{
    position: relative;
    align-items: center;
    justify-content: center;
    background-color: #2782D7;
    color: #FFFFFF;
    font-size: 36rpx;
  }
  .demo-text-2:before{
    content: 'B';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .demo-text-3{
    position: relative;
    align-items: center;
    justify-content: center;
    background-color: #F1F1F1;
    color: #353535;
    font-size: 36rpx;
  }
  .demo-text-3:before{
    content: 'C';
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .btn-area {
    margin-top: 20rpx;
    box-sizing: border-box;
    width: 100%;
    padding: 0 30rpx;
  }
```

(6) 添加一个名为 swiper 的编译模式,并将 5_3_swiper 设置为启动页面,使用新的 swiper 编译模式编译项目并在模拟器中预览效果。

【相关知识】

swiper 组件和前面学到的 scroll-view 组件不一样的是,swiper 组件是一次滑动一项,而 scroll-view 组件里的内容可以连续滑动。因此在我们的 swiper 示例页面中,除了一个<swiper></swiper>标签外,里面还包含表示滑块项目的<swiper-item></swiper-item>标签。

swiper 组件的相关属性见表 5-4。

表 5-4　swiper 组件属性

属　性　名	类　　型	默　认　值	说　　明	最低版本
indicator-dots	Boolean	false	是否显示面板指示点	
indicator-color	Color	rgba(0, 0, 0, .3)	指示点颜色	1.1.0
indicator-active-color	Color	#000000	当前选中的指示点颜色	1.1.0
autoplay	Boolean	false	是否自动切换	
current	Number	0	当前所在滑块的 index	
current-item-id	String	""	当前所在滑块的 item-id，不能与 current 被同时指定	1.9.0
interval	Number	5000	自动切换时间间隔	
duration	Number	500	滑动动画时长	
circular	Boolean	false	是否采用衔接滑动	
vertical	Boolean	false	滑动方向是否为纵向	
previous-margin	String	"0px"	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"	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	同时显示的滑块数量	1.9.0
skip-hidden-item-layout	Boolean	false	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
bindchange	EventHandle		current 改变时会触发 change 事件，event.detail = {current: current, source: source}	
bindanimationfinish	EventHandle		动画结束时会触发 animationfinish 事件，event.detail 同上	1.9.0

在 swiper 组件中，只可以放置 swiper-item 组件，反之，swiper-item 组件也只能放置在 swiper 组件中，而且其宽高会自动设置为 100%。swiper-item 组件的属性说明见表 5-5。

表 5-5　swiper-item 组件属性说明

属　性　名	类　　型	默　认　值	说　　明	最低版本
item-id	String	""	该 swiper-item 的标识符	1.9.0

在本例中，我们在< swiper ></swiper >标签里面放置了三个< swiper-item ></swiper-item >标签，分别表示三个色块。在< swiper ></swiper >标签中，swiper 组件用于确定是否显示面板指示点的属性 indicator-dots 的值绑定到了后端的 indicatorDots 变量上。在 5_3_swiper.js 文件中的 data 对象中，indicatorDots 被赋值为 true，也就是默认显示指示点。我们为按钮的单击事件绑定了处理函数 changeIndicatorDots，每次单击按钮，changeIndicatorDots 函数便会将当前的 indicatorDots 变量值取反，实现通过单击按钮切换指示点显示状态的效果。

使用 swiper 组件，需要注意以下几点。

（1）swiper 组件里只能放置 swiper-item 组件，swiper-item 组件也只能被放置在 swiper 组件中。

（2）swiper 组件的 change 事件返回的 detail 里面，source 字段表示导致变更的原因。source 字段的可能值如下。

① autoplay：自动播放导致 swiper 发生变化。

② touch：用户滑动引起 swiper 发生变化。

③ ""：其他原因将用空字符串表示。

（3）如果在 bindchange 的事件回调函数中使用 setData 改变 current 值，则有可能导致 setData 被不停地调用，因而通常情况下请在改变 current 值前检测 source 字段来判断是否是由于用户触摸引起的。

5.4 可移动视图容器 movable-view 和 movable-area

【任务要求】

使用 movable-view 组件和 movable-area 组件新建如图 5-10 所示页面，分别实现滑块的横向移动，纵向移动，以及任意移动。实现按钮的单击移动到固定位置的功能，并在任意移动的滑块 C 中，绑定 change 事件，在控制台观察滑块移动事件的输出。

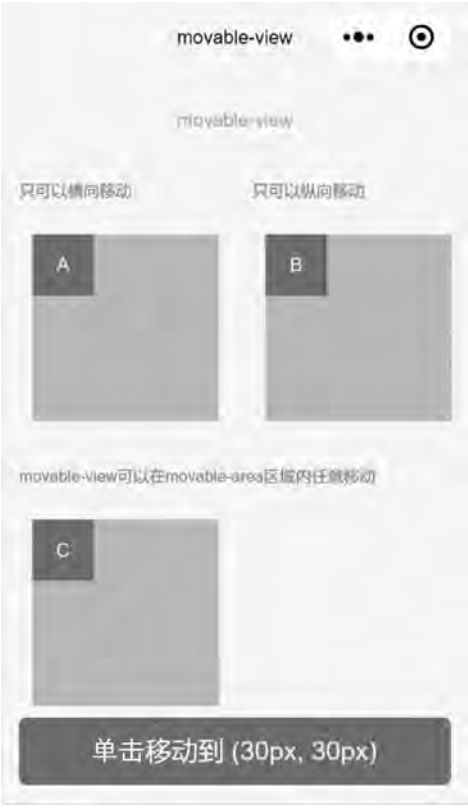

图 5-10　movable-view 组件示例

【任务分析】

本任务使用到的是可移动视图容器 movable-view 和 movable-area。movable-view 表示可移动的组件/视图，movable-area 表示可移动的范围。在本例中，需要限制可以移动的方向，可以通过设置 movable-view 组件的 direction 属性值来实现。需要实现单击按钮将滑块移动到指定位置的功能，可以通过监听按钮单击事件，动态修改 movable-view 的 x、y 属性值来实现。

【任务操作】

(1) 打开示例小程序项目，在 app.json 文件的 pages 数组中新增页面“pages/Chapter_5/5_4_movable-view/5_4_ movable-view”，单击“编译”按钮，生成 5_4_ movable-view 页面所需的文件。

(2) 修改 5_4_movable-view.json 文件内容为如下代码，使页面的窗口标题显示 movable-view。

```
{
  "navigationBarTitleText": "movable - view"
}
```

(3) 修改文件 5_4_movable-view.wxml 的内容为如下代码，完成页面元素的布局。

```
<!-- pages/Chapter_5/5_4_movable - view/5_4_movable - view.wxml -->
<view class = "container">
  <view class = "page - head">
    <view class = "page - head - title"> movable - view </view>
    <view class = "page - head - line"></view>
  </view>

  <view class = "page - body">
    <view class = "wrap">
      <view class = "page - section">
        <view class = "page - section - title top">只可以横向移动</view>
        < movable - area >
          < movable - view direction = "horizontal"> A </movable - view >
        </movable - area >
      </view>

      <view class = "page - section">
        <view class = "page - section - title top">只可以纵向移动</view>
        < movable - area >
          < movable - view direction = "vertical"> B </movable - view >
        </movable - area >
      </view>
    </view>

    <view class = "page - section">
      <view class = "page - section - title"> movable - view 可以在 movable - area 区域内任意移动</view>
      <movable - area>
        < movable - view x = "{{x}}" y = "{{y}}" direction = "all" bindchange = "onChange"> C
```

```
</movable-view>
      </movable-area>
    </view>
    <view class="btn-area">
      <button bindtap="tap" class="page-body-button" type="primary">单击移动到(30px, 30px)</button>
    </view>
  </view>
</view>
```

(4) 修改文件 5_4_movable-view.wxss 的内容为如下代码,完成页面的样式调整。

```
/* pages/Chapter_5/5_4_movable-view/5_4_movable-view.wxss */
page {
  background-color: #F8F8F8;
  height: 100%;
  font-size: 32rpx;
  line-height: 1.6;
}
button{
  margin-top: 20rpx;
  margin-bottom: 20rpx;
}
.container {
  display: flex;
  flex-direction: column;
  min-height: 100%;
  justify-content: space-between;
  font-size: 32rpx;
  font-family: -apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
.wrap {
  display: flex;
  flex-direction: row;
  min-height: 100%;
}
.page-head{
  padding: 60rpx 50rpx 50rpx;
  text-align: center;
}
.page-head-title{
  display: inline-block;
  padding: 0 40rpx 20rpx 40rpx;
  font-size: 32rpx;
  color: #BEBEBE;
}
.page-head-line{
  margin: 0 auto;
  width: 150rpx;
  height: 2rpx;
  background-color: #D8D8D8;
```

```
}
.page-body {
  width: 100%;
  flex-grow: 1;
  overflow-x: hidden;
}
.page-section{
  width: 100%;
  margin-bottom: 20rpx;
}
.page-section-title{
  margin-top: 50rpx;
  font-size: 28rpx;
  color: #999999;
  margin-bottom: 10rpx;
  padding-left: 30rpx;
  padding-right: 30rpx;
}
.page-section-title.top {
  margin-top: 0;
}
movable-view {
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100rpx;
  width: 100rpx;
  background: #1AAD19;
  color: #fff;
}
movable-area {
  height: 300rpx;
  width: 300rpx;
  margin: 50rpx 0rpx 0 50rpx;
  background-color: #ccc;
  overflow: hidden;
}
.btn-area {
  margin-top: 20rpx;
  box-sizing: border-box;
  width: 100%;
  padding: 0 30rpx;
}
```

(5) 在5_4_movable-view.js文件中,设定属性x,y的初始值为0,新增用于处理按钮单击事件的tap()函数以及处理滑块移动事件的onChange()函数。修改完成后的5_4_movable-view.js文件内容如下。

```
// pages/Chapter_5/5_4_movable-view/5_4_movable-view.js
Page({
  data: {
```

```
    x: 0,
    y: 0,
  },
  onLoad: function (options) {},
  onReady: function () {},
  onShow: function () {},
  onHide: function () {},
  onUnload: function () {},
  onPullDownRefresh: function () {},
  onReachBottom: function () {},
  onShareAppMessage: function () {},
  tap() {
    this.setData({
      x: 30,
      y: 30
    })
  },
  onChange(e) {
    console.log(e.detail)
  }
})
```

(6) 新建名为 movable-view 的编译模式,并设置“pages/Chapter_5/5_3_swiper/5_3_swiper”为启动页面。使用 movable-view 编译模式编译项目,观察模拟器显示效果。

(7) 使用鼠标按住并移动滑块“C”,打开控制台,可以看到 onChange()函数被执行并输出如图 5-11 所示 change 事件的详情。

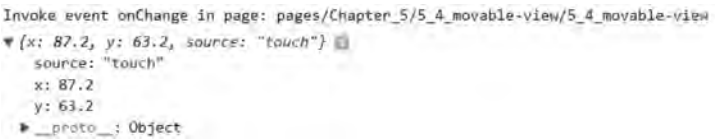

图 5-11 onChange()函数输出结果

【相关知识】

movable-view 表示可移动的视图容器,从基础库 1.2.0 开始支持,该容器可以在页面中拖曳滑动。其属性值说明见表 5-6。

表 5-6 movable-view 组件属性说明

属 性 名	类 型	默认值	说 明	最低版本
direction	String	none	movable-view 的移动方向,属性值有 all、vertical、horizontal、none	
inertia	Boolean	false	movable-view 是否带有惯性	
out-of-bounds	Boolean	false	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,如果 x 的值不在可移动范围内,会自动移动到可移动范围;改变 x 的值会触发动画	

续表

属性名	类型	默认值	说明	最低版本
y	Number / String		定义 y 轴方向的偏移，如果 y 的值不在可移动范围内，会自动移动到可移动范围；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，滑动越快停止；必须大于 0，否则会被设置成默认值	
disabled	Boolean	false	是否禁用	1.9.90
scale	Boolean	false	是否支持双指缩放，默认缩放手势生效区域是在 movable-view 内	1.9.90
scale-min	Number	0.5	定义缩放倍数最小值	1.9.90
scale-max	Number	10	定义缩放倍数最大值	1.9.90
scale-value	Number	1	定义缩放倍数，取值范围为 0.5～10	1.9.90
animation	Boolean	true	是否使用动画	2.1.0
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)	1.9.90
bindscale	EventHandle		缩放过程中触发的事件，event.detail = {x: x, y: y, scale: scale}，其中 x 和 y 字段在 2.1.0 之后开始支持返回	1.9.90

除了基本事件外，movable-view 提供了两个特殊事件，说明见表 5-7。

表 5-7　movable-view 特殊事件

类型	触发条件	最低版本
htouchmove	初次手指触摸后移动为横向的移动，如果 catch 此事件，则意味着 touchmove 事件也被 catch	1.9.90
vtouchmove	初次手指触摸后移动为纵向的移动，如果 catch 此事件，则意味着 touchmove 事件也被 catch	1.9.90

movable-area 表示 movable-view 可移动的区域。其属性说明见表 5-8。

表 5-8　movable-area 属性说明表

属性名	类型	默认值	说明	最低版本
scale-area	Boolean	false	当里面的 movable-view 设置为支持双指缩放时，设置此值可将缩放手势生效区域修改为整个 movable-area	1.9.90

movable-view 组件必须被包含在 movable-area 组件中，并且必须是直接子节点，否则便没有移动效果。当 movable-view 小于 movable-area 时，movable-view 的移动范围是在 movable-area 内；当 movable-view 大于 movable-area 时，movable-view 的移动范围必须包含 movable-area(x 轴方向和 y 轴方向分开考虑)。

在本例中，三个可移动区域均是 movable-view 小于 movable-area，因此带有文本的色块也就只能在限定区域内移动。滑块“A”通过设定其 direction 属性值为 horizontal，限制了其只能横向移动；同理，滑块“B”的 direction 属性值被设置为 vertical，因此只能纵向滑动。滑块“C”的 direction 属性值为 all，表示不限制滑动方向，即可以在 movable-area 组件区域内任意移动。同时，滑块“C”的 x 属性，也就是横坐标值绑定到了后端变量 x 上；y 属性，也就是纵坐标值，绑定到了后端变量 y 上，并且在 5_4_movable-view.js 文件的 data 对象中，将 x,y 初始化为 0，也就是顶齐 movable-area 区域的左上顶点显示。按钮的单击事件处理函数 tap()则通过直接将 x,y 的值设置为 30，实现了单击移动滑块“C”到指定位置的功能。

在使用鼠标按住并拖动滑块“C”的过程中，可以看到 change 事件的输出(如图 5-11 所示)，e.detail 包含滑块的横、纵坐标信息，还有一个 source 字段表示产生改变的原因。当我们使用鼠标拖动时，可以看到 source 的值为“touch”，而当我们单击按钮直接将其定位到坐标为(30,30)的点时，可以看到控制台的输出如图 5-12 所示。

```
Invoke event onChange in page: pages/Chapter_5/5_4_movable-view/5_4_movable-view
▶{x: 29.7, y: 29.7, source: ""}
Invoke event onChange in page: pages/Chapter_5/5_4_movable-view/5_4_movable-view
▶{x: 29.8, y: 29.8, source: ""}
Invoke event onChange in page: pages/Chapter_5/5_4_movable-view/5_4_movable-view
▶{x: 29.9, y: 29.9, source: ""}
Invoke event onChange in page: pages/Chapter_5/5_4_movable-view/5_4_movable-view
▶{x: 30, y: 30, source: ""}
```

图 5-12　使用按钮设置位置触发 change 事件输出

控制台的输出展示了滑块“C”从位置(0,0)移动到(30,30)的全过程，由于篇幅限制，图 5-12 只截取了最后的四次输出。可以看到和图 5-11 展示的输出不同的是，图 5-12 的 source 字段为空字符串，因为这里的移动，是通过 tap()函数里面的 setData()直接改变 x,y 的值实现的，因此为空字符串。

使用 movable-view 和 movable-area 组件，还有以下几点注意事项。

(1) movable-view 必须设置 width 和 height 属性，不设置则默认为 10px；

(2) movable-view 默认为绝对定位，top 和 left 属性为 0px；

(3) movable-area 必须设置 width 和 height 属性，不设置则默认为 10px。

5.5　cover-view 组件和 cover-image 组件

【任务要求】

新建如图 5-13 所示页面，使用 video 标签在页面上放置一个视频，视频地址为 http://t.cn/RIt6r8j。在视频上面使用 cover-view 组件和 cover-image 组件放置三个由图片组成的控件，从左至右分别是播放、暂停以及停止，实现对视频播放的控制。

图 5-13　cover-view 和 cover-image 任务示例

【任务分析】

本次任务主要是针对 cover-view 和 cover-image 组件的应用。这两个组件都是可以覆盖在其他组件之上的。在本次任务中，涉及视频组件的使用，包括对视频的控制操作，这部分内容可以参考第 8 章以及第 12 章内容，本次任务视频控制部分将不作重点讲解。

【任务操作】

(1) 打开示例小程序项目，在 app.json 文件的 pages 数组中新增页面“pages/Chapter_5/5_5_cover-view/5_5_ cover-view”，单击“编译”按钮，生成 5_5_ cover-view 页面所需的文件。

(2) 在示例小程序项目的根目录下新建一个 image 文件夹，用于放置图片文件。在阿里巴巴矢量图标库(http://www.iconfont.cn/)中，分别以 play、pause、stop 为关键字搜索图标并下载，并将下载好的图片放置到 image 文件夹中。完成后的小程序文件目录结构如图 5-14 所示。

(3) 在文件 5_5_cover-view.json 中写入如下代码，配置窗口显示为“cover-view”。

```
{
  "navigationBarTitleText": "cover - view"
}
```

(4) 在文件 5_5_cover-view.wxml 中写入如下代码，完成页面元素的排布。

```
<! -- pages/Chapter_5/5_5_cover - view/5_5_cover - view.wxml -- >
< view class = "container">
  < view class = "page - head">
    < view class = "page - head - title"> cover - view </view >
    < view class = "page - head - line"></view >
```

图 5-14　image 文件夹

```
  </view>

  <view class="page-body">
    <view class="page-section page-section-gap">
      <video id="myVideo" src="http://t.cn/RIt6r8j" controls="true" event-model="bubble">
        <cover-view class="controls">
          <cover-view class="play" bindtap="play">
            <cover-image class="img" src="../../../image/play.png" />
          </cover-view>
          <cover-view class="pause" bindtap="pause">
            <cover-image class="img" src="../../../image/pause.png" />
          </cover-view>
          <cover-view class="stop" bindtap="stop">
            <cover-image class="img" src="../../../image/stop.png" />
          </cover-view>
        </cover-view>
      </video>
    </view>
  </view>

</view>
```

(5) 在 5_5_cover-view.wxss 文件中写入如下代码,完成页面样式的调整。

```
/* pages/Chapter_5/5_5_cover-view/5_5_cover-view.wxss */
page {
```

```
  background-color: #F8F8F8;
  height: 100%;
  font-size: 32rpx;
  line-height: 1.6;
}
.container {
  display: flex;
  flex-direction: column;
  min-height: 100%;
  justify-content: space-between;
  font-size: 32rpx;
  font-family: -apple-system-font,Helvetica Neue,Helvetica,sans-serif;
}
.page-head{
  padding: 60rpx 50rpx 80rpx;
  text-align: center;
}
.page-head-title{
  display: inline-block;
  padding: 0 40rpx 20rpx 40rpx;
  font-size: 32rpx;
  color: #BEBEBE;
}
.page-head-line{
  margin: 0 auto;
  width: 150rpx;
  height: 2rpx;
  background-color: #D8D8D8;
}
.page-body {
  width: 100%;
  flex-grow: 1;
  overflow-x: hidden;
}
.page-section{
  width: 100%;
  margin-bottom: 20rpx;
}
.page-section-title{
  margin-top: 50rpx;
  font-size: 28rpx;
  color: #999999;
  margin-bottom: 10rpx;
  padding-left: 30rpx;
  padding-right: 30rpx;
}
.page-section-gap{
  box-sizing: border-box;
  padding: 0 30rpx;
}
.controls {
```

```
    position: relative;
    top: 50%;
    height: 50px;
    margin-top: -25px;
    display: flex;
  }
  .play,.pause,.stop {
    flex: 1;
    height: 100%;
  }
  .img {
    width: 40px;
    height: 40px;
    margin: 5px auto;
  }
  video{
    width: 100%
  }
```

(6) 在 5_5_cover-view.js 文件中,新建 play()函数、pause()函数、stop()函数实现对视频播放的控制。

```
// pages/Chapter_5/5_5_cover-view/5_5_cover-view.js
Page({
  data: {},
  onLoad: function (options) {},
  onReady: function () {
    this.videoCtx = wx.createVideoContext('myVideo')
  },
  onShow: function () {},
  onHide: function () {},
  onUnload: function () {},
  onPullDownRefresh: function () {},
  onReachBottom: function () {},
  onShareAppMessage: function () {},
  play() {
    this.videoCtx.play()
  },
  pause() {
    this.videoCtx.pause()
  },
  stop() {
    this.videoCtx.stop()
  }
})
```

(7) 新建一个名为 cover-view 的编译模式,设置“pages/Chapter_5/5_5_cover-view/5_5_cover-view”为启动页面。使用 cover-view 编译模式编译项目并在模拟器中观察页面效果。

【相关知识】

cover-view 和 cover-image 组件均是从基础库 1.4.0 开始支持。cover-view 表示覆盖在原生组件之上的文本视图，可覆盖的原生组件包括 map、video、canvas、camera、live-player 和 live-pusher，只支持嵌套 cover-view 和 cover-image，可在 cover-view 中使用 button。cover-image 表示覆盖在原生组件之上的图片视图，可覆盖的原生组件同 cover-view，支持嵌套在 cover-view 里。

cover-view 组件的属性说明见表 5-9。

表 5-9　cover-view 组件属性说明

属性名	类　型	默认值	说　明	最低版本
scroll-top	Number/String		设置顶部滚动偏移量，仅在设置了 overflow-y：scroll 成为滚动元素后生效（单位为 px，2.4.0 起支持 rpx）	2.1.0

cover-image 组件的属性说明见表 5-10。

表 5-10　cover-image 组件属性说明

属性名	类　型	默认值	说　明	最低版本
src	String		图标路径，支持临时路径、网络地址（1.6.0 起支持）、云文件 ID（2.2.3 起支持）。暂不支持 base64 格式	
bindload	EventHandle		图片加载成功时触发	2.1.0
binderror	EventHandle		图片加载失败时触发	2.1.0

在本次任务中，在< video ></video >标签中嵌套了一个 cover-view 组件，在这个 cover-view 组件中，又嵌套了三个 cover-view 组件，其中，每个 cover-view 组件都嵌套了一个 cover-image 组件，用于放置三个控件图片。在三个 cover-view 组件中，每个组件都绑定了 tap 事件，对应的处理函数 play()，pause()和 stop()实现了对视频播放的控制。

使用 cover-view 组件和 cover-image 组件需要注意以下几点。

（1）基础库 2.2.4 起支持 touch 相关事件，也可使用 hover-class 设置单击态；

（2）基础库 2.1.0 起支持设置 scalerotate 的 CSS 样式，包括 transition 动画；

（3）基础库 1.9.90 起 cover-view 支持 overflow：scroll，但不支持动态更新 overflow；

（4）基础库 1.9.90 起最外层 cover-view 支持 position：fixed；

（5）基础库 1.9.0 起支持插在 view 等标签下。在此之前只可嵌套在原生组件 map、video、canvas 和 camera 内，避免嵌套在其他组件内；

（6）基础库 1.6.0 起支持 csstransition 动画，transition-property 只支持 transform（translateX，translateY）与 opacity；

（7）基础库 1.6.0 起支持 cssopacity；

（8）事件模型遵循冒泡模型，但不会冒泡到原生组件；

（9）文本建议都套上 cover-view 标签，避免排版错误；

（10）只支持基本的定位、布局、文本样式，不支持设置单边的 border、background-

image、shadow、overflow:visible 等；

（11）建议子节点不要溢出父节点；

（12）默认设置的样式有：white-space:nowrap;line-height:1.2;display:block；

（13）自定义组件嵌套 cover-view 时，自定义组件的 slot 及其父节点暂不支持通过 wx:if 控制显隐，否则会导致 cover-view 不显示。

练　习　题

1. 请使用表 5-1 中的 justify-content 属性，通过为其设置不同的值，实现如图 5-15 所示的显示效果。

2. 请使用表 5-1 中的 align-items 属性，通过为其设置不同的值，实现如图 5-16 所示的显示效果。

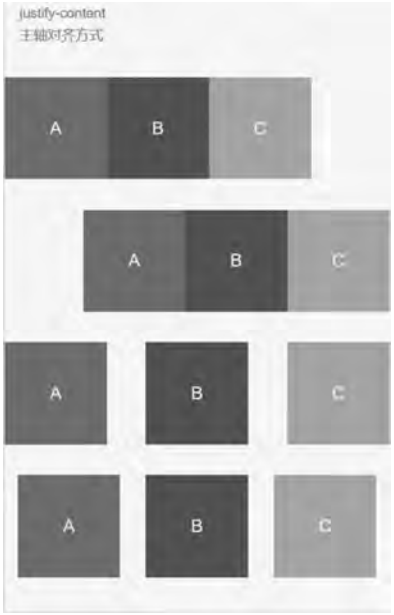

图 5-15　主轴对齐方式练习

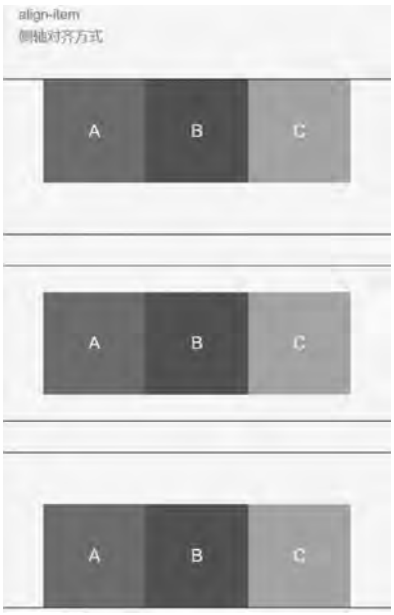

图 5-16　侧轴对齐方式练习

3. 在页面上新增两个按钮，一个按钮实现单击一次，色块向下移动 10px 的功能，另一个按钮实现单击一次，滑动到下一个色块的功能（如图 5-17 所示）。提示：需要使用到表 5-3 中的 scroll-top 和 scroll-into-view 属性，然后为按钮设置监听单击事件，动态改变上述两个属性的值。

4. 新建如图 5-18 所示页面，要求滑块为纵向滑动，默认显示指示点，同时实现四个按钮的相关功能。

5. 新建一个如图 5-19 所示页面，放置 A、B、C 三个可移动区域和滑块。其中，区域 A 的 movable-view 大于 movable-area，对 A 色块设置渐变色便于观察 movable-view 和 movable-area 的边界；对 B 色块的属性值进行设置使得 movable-view 边界可以超出 movable-area 边界；对 C 色块绑定 change 事件和 scale 事件，新增一个按钮实现单击放大 C 色块的功能，并在控制台观察 C 色块 change 事件和 scale 事件的输出详情。

图 5-17　通过按钮控制滚动

图 5-18　swiper 组件练习示例

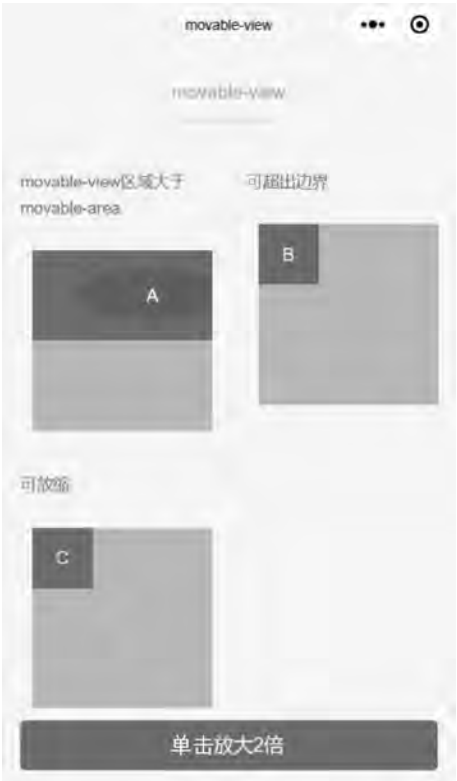

图 5-19　movable-view 练习示例

6. 新建如图 5-20 所示页面，在地图组件上覆盖 A、B、C 三个色块，并给色块设置一定的透明度。

图 5-20　cover-view 练习