第 3 章

微信小程序组件

微信小程序框架为开发者提供了一系列的基础组件，这些原生组件让小程序具有良好的用户体验，同时也方便开发者快速开发。

本章主要目标

- 了解小程序组件的含义；
- 熟练掌握常见的容器组件、内容组件、表单组件、导航组件、媒体组件和地图组件的属性以及用法；
- 综合运用小程序组件完成问卷调查项目的设计与开发。

3.1 组件概述

组件是视图层基本的组成单元，具备 UI 风格样式以及特定的功能效果。当打开某款小程序后，界面中的图片、文字等元素都需要使用组件，小程序组件使用灵活，组件之间通过相互嵌套进行界面设计，开发者可以通过组件的选择和样式属性设计出不同的界面效果。一个组件包括开始标签和结束标签，属性用来装饰这个组件的样式。

其语法格式如下：

```
<标签名称 属性 = "值">
内容
</标签名称>
```

示例代码如下：

```
< button class = "btn">我是按钮组件</button >
```

上述代码用< button ></button >表示一个按钮组件，在< button >标签中通过 class="btn" 为< button >组件添加样式 btn。小程序目前提供的通用属性如表 3.1 所示。

表 3.1　小程序组件通用属性

属　性　名	类　　型	说　　明	备　　注
id	string	组件的唯一标识	在当前界面中用 id 值标识唯一的组件，并且不能有两个及以上 id 值同名
class	string	组件的样式类	为一个或多个组件设置样式类
style	string	组件的内联样式	动态设置内联样式
hidden	boolean	组件的显示/隐藏	组件均默认为显示状态
data-*	any	自定义属性	当组件触发事件时会附带将该属性和值发送给对应的事件处理函数
bind*/catch*	eventHandler	组件的事件	为组件绑定/捕获事件

3.2　视图容器组件

视图容器(View Container)组件用于排版页面为其他组件提供载体。视图容器有view、scroll-view 和 swiper 3 种。

3.2.1　view

view 容器是页面中最基本的容器组件，通过高度和宽度来定义容器大小。< view >相当于 HTML 中的< div >标签，是一个页面中最外层的容器，能够接受其他组件的嵌入，例如，多个 view 容器的嵌套。view 容器可以通过 flex 布局定义内部项目的排列方式(详见第4章 flex 布局)。其属性如表 3.2 所示。

表 3.2　＜view＞组件属性

属　性　名	类型	默认值	说　　明
hover	boolean	false	是否启动点击态
hover-class	string	none	按住容器后的样式。当属性设置为 hover-class＝"none"时，没有点击效果
hover-stop-propagation	boolean	false	指定是否阻止本容器的祖先节点出现点击态
hover-start-time	number	50	按住容器后多久出现点击态，单位为 ms
hover-stay-time	number	400	手指离开后点击态的保留时长，单位为 ms

例 3-1　本例设计了两组父子 view 容器的点击态，第一组父子 view 容器中子 view 容器不阻止点击态向父容器传递，第二组父子 view 容器中子 view 容器阻止点击态向父容器传递，程序运行效果如图 3.1 所示。

视频讲解

pages/view/view.wxml 文件代码如下：

```
< view class = "demo - box">
  < view class = "title"> 1. view 小案例</view >
  < view class = "title">(1)不阻止父容器的 view - hover </view >
```

```
  <view class="view-parent" hover-class="view-hover">我是父类容器
    <view class="view-son" hover-class="view-hover">我是子类容器</view>
  </view>
  <view class="title">(2)阻止父容器的 view-hover </view>
  <view class="view-parent" hover-class="view-hover">我是父类容器
    <view class="view-son" hover-class="view-hover" hover-stop-propagation hover-start-time="3000" hover-stay-time="4000">我是子类容器</view>
  </view>
</view>
```

(a) 页面初始效果　　(b) 点击第1组子容器　　(c) 点击第2组子容器3s后

图 3.1　组件 view 小案例

pages/view/view.wxss 文件代码如下：

```
.view-parent {
  width: 100%;
  height: 350rpx;
  background-color: pink;
  text-align: center;
}
.view-son {
  width: 50%;
  height: 200rpx;
  background-color: skyblue;
  margin: 20rpx auto;
  text-align: center;
}
.view-hover {
  background-color: red;
}
```

app.wxss 文件代码如下：

```
.demo-box {
  padding: 20rpx; margin: 20rpx 60rpx; border: 1rpx solid gray;
}
.title {
  display: flex;
  flex-direction: row;
  margin: 20rpx;
  justify-content: center;
}
```

app.wxss 文件中代码为公共样式，用于设置页面的布局以及标题样式，在本章所有案例中均相同，在后面的案例中省略。

【代码讲解】 本例在 view.wxml 文件中放置两组<view>容器，在 app.wxss 文件中设置父容器背景色为浅红色，子容器背景色为浅蓝色，通过 hover-class="view-hover"为标签增加属性，点击态均设置为点击后背景色更新为红色，第一组不阻止点击态传递给父容器，在第二组子类容器中通过 hover-stop-propagation 来阻止点击态传递给父容器，并设置属性 hover-start-time="3000"，hover-stay-time="4000"，当点击子容器时，3s 后出现点击状态，当手指松开 4s 后，子容器背景色变为初始颜色。

图 3.1(a)为页面初始效果；图 3.1(b)为点击第 1 组的子容器后，父子容器背景色均变为红色；图 3.1(c)为点击第 2 组的子容器后，仅有子容器背景色变为红色。

3.2.2　scroll-view

scroll-view 容器为可滚动的视图容器，允许用户通过手指在容器上滑动来改变显示区域，常见的滑动方向有水平滑动和垂直滑动。其属性如表 3.3 所示。

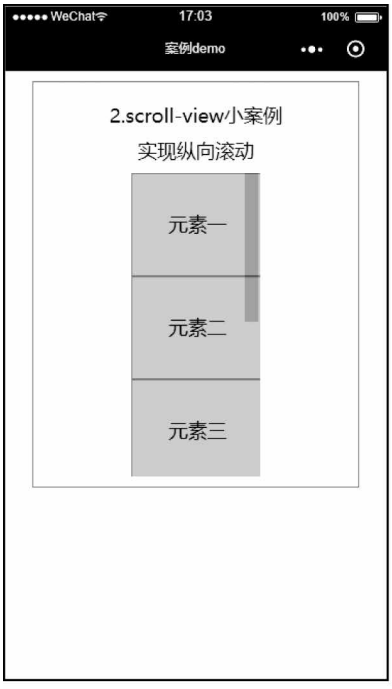
表 3.3　<scroll-view>组件属性

属　性　名	类　　型	默认值	说　　明
scroll-x	boolean	false	允许横向滑动
scroll-y	boolean	none	允许纵向滑动
upper-threshold	number	50	距顶部/左边多远时(单位：px)，触发 scrolltoupper 事件
lower-threshold	number	50	距顶部/右边多远时(单位：px)，触发 scrolltolower 事件
scroll-top	number		设置纵向滚动条位置
scroll-left	number		设置横向滚动条位置
scroll-into-view	string		值应为某子元素 id。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 下单击顶部状态、Android 双击标题栏滚动条返回顶部，仅支持纵向
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
bingscroll	eventhandle		滚动时触发，event.detail = {scrollLeft，scrollTop，scrollHeight，scrollWidth，deltaX，deltaY}

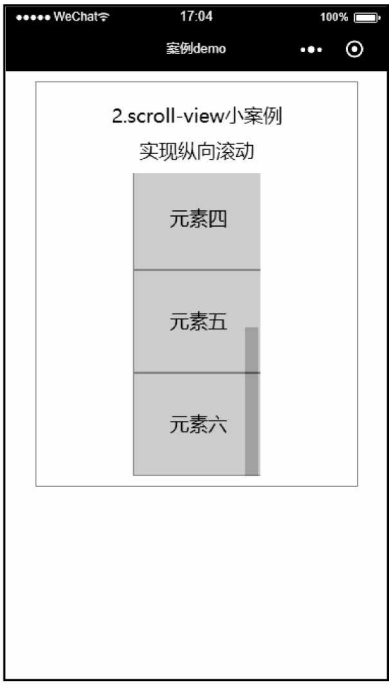
注意：在使用纵向滚动时，需要为< scroll-view >设置一个固定宽度。

例 3-2 本例设计一个纵向 scroll-view 组件，运行效果如图 3.2 所示。

视频讲解



(a) 页面初始效果



(b) scroll-view滚动后效果

图 3.2 组件 scroll-view 小案例

pages/scroll-view/scroll-view.wxml 文件代码如下：

```
< view class = "demo - box">
< view class = "title"> 2. scroll - view 小案例</view >
< view class = "title">实现纵向滚动</view >
< scroll - view scroll - y >
< view class = "scroll - item - y">元素一</view >
< view class = "scroll - item - y">元素二</view >
< view class = "scroll - item - y">元素三</view >
< view class = "scroll - item - y">元素四</view >
< view class = "scroll - item - y">元素五</view >
< view class = "scroll - item - y">元素六</view >
</scroll - view >
</view >
```

pages/scroll-view/scroll-view.wxss 文件代码如下：

```
scroll - view {
  height: 600rpx; width: 250rpx; margin: 0 auto;
}
. scroll - item - y {
  height: 200rpx; line - height: 200rpx;
  text - align: center; background - color: skyblue; border: 1px solid gray;
```

```
}
```

【代码讲解】 本例在 scroll-view.wxml 文件中放置< scroll-view >组件，通过设置属性 scroll-y，允许组件上下滑动，在 scroll-view.wxss 文件中设置其高度为 600rpx，使得 scroll-view 组件能够纵向滚动，在< scroll-view >中嵌套 6 组< view >用于显示滚动效果，内部元素宽度均为 250rpx。

图 3.2(a)为页面初始效果；图 3.2(b)为< scroll-view >组件滑动到底部后的效果。

在图 3.3 中，虚线框是< scroll-view >组件在京东小程序分类页中的应用。

图 3.3　scroll-view 组件应用实例

3.2.3　swiper

< swiper >组件为滑块视图容器，通常用于图片之间的切换播放，被形象地称为轮播图。其属性如表 3.4 所示。

表 3.4　< swiper >组件属性

属　性　名	类　　型	默　认　值	说　　明
indicator-dots	boolean	false	是否显示面板指示点
indicator-color	color	rgba(0,0,0,0.3)	指示点颜色
indicator-active-color	color	#000000	当前选择的指示点颜色

续表

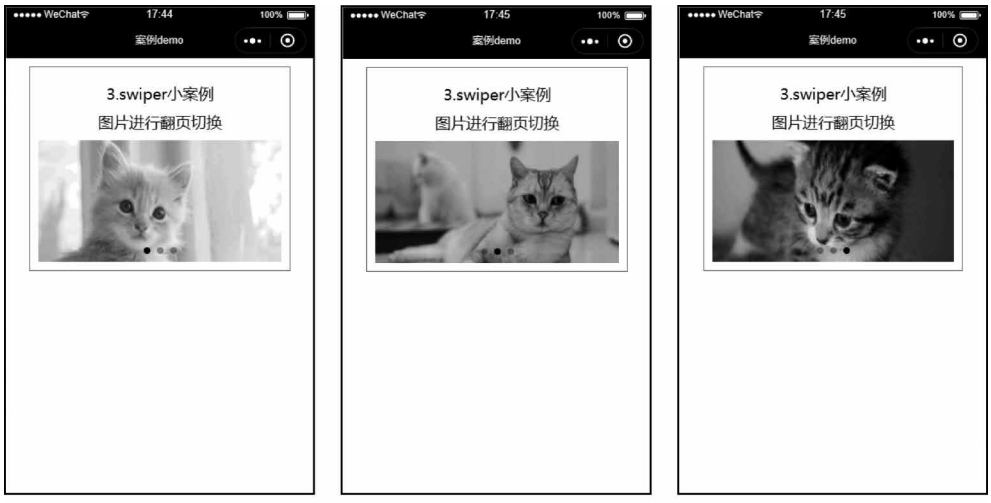
属 性 名	类 型	默 认 值	说 明
autoplay	boolean	false	是否自动切换
current	number	0	当前所在滑块的 index
current-item-id	string	" "	当前所在滑块的 item-id，不能与 current 被同时指定
interval	number	5000	自动切换时间间隔，单位：ms
duration	number	500	滑动动画时长，单位：ms
circular	boolean	false	是否采用衔接滑动
vertical	boolean	false	滑动方向是否为纵向
previous-margin	string	"0px"	前边距，可用于露出前一项的一小部分，接受 px 和 rpx 值
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
display-multiple-items	number	1	同时显示的滑块数量
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
bindchange	eventhandle		current 改变时会触发 change 事件，event.detail = {current：current，source：source}
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

例 3-3 swiper 组件小案例，运行效果如图 3.4 所示。

视频讲解

pages/swiper/swiper.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">3.swiper 小案例</view>
  <view class="title">图片进行翻页切换</view>
  <swiper indicator-dots autoplay interval="3000">
    <swiper-item>
      <image src="/images/cat1.jpg"></image>
    </swiper-item>
    <swiper-item>
      <image src="/images/cat2.jpg"></image>
    </swiper-item>
    <swiper-item>
      <image src="/images/cat3.jpg"></image>
    </swiper-item>
  </swiper>
</view>
```



(a) 页面初始效果　　(b) 切换到第2张照片　　(c) 切换到第3张照片

图 3.4　swiper 组件小案例

pages/swiper/swiper.wxss 文件代码如下：

```
swiper {
  height: 350rpx;
}
```

【代码讲解】 本例在 swiper.wxml 文件中放置<swiper>组件，设置属性 autoplay 允许自动切换图片，设置属性 interval="3000"，图片每隔 3s 发生一次切换，属性 indicator-dots 用于显示面板指示点，<swiper>组件中嵌套 3 组<swiper-item>，swiper 容器的高度设置为 300rpx。

图 3.4(a)为页面初始效果，此时默认显示第一张图片；图 3.4(b)和图 3.4(c)分别显示第二张照片和第三张照片，照片数据来自本地，保存在 images 文件夹下。

在图 3.5 中，虚线框是<swiper>组件在携程小程序中的应用。

图 3.5　swiper 组件应用实例

3.3　基础内容组件

3.3.1　icon

<icon>为图标组件，常用于页面装饰，开发者可以自定义其类型、大小和颜色。其属性

如表 3.5 所示。

表 3.5 ＜icon＞组件属性

属性名	类　　型	默认值	说　　明
type	string	false	icon 的类型，有效值：success，success_no_circle，info，warn，waiting，cancel，download，search，clear
size	number/string	23px	icon 的大小，单位：px(基础库 2.4.0 起支持 rpx)
color	color	#000000	icon 的颜色，同 CSS 的 color

例如，自定义一个绿色、40px 大小的 success 图标。

WXML 中的代码如下：

```
< icon type = "success" size = "40" color = "green" />
```

如果有多个图标需要批量生成，利用 wx:for 循环精简代码，在 JS 文件的 data 中存放数据，然后在 WXML 文件中使用< block >标签进行列表渲染。

批量生成不同大小的 success 图标的示例代码如下。

WXML 中的代码如下：

```
< view >
  < block wx:for = "{{iconSize}}">
    < icon type = "success" size = "{{item}}" />
  </block >
</view >
```

JS 文件代码如下：

```
Page({
  data: {
    iconSize: ["20", "25", "30"]
  }
})
```

上述代码生成的图标大小分别为 20rpx、25rpx 和 30rpx。

例 3-4　icon 组件小案例，运行效果如图 3.6 所示。

pages/icon/icon.wxml 文件代码如下：

视频讲解

```
< view class = "demo - box">
  < view class = "title"> 4. icon 小案例</view >
  < view class = "title">(1)实现大小变化</view >
  < block wx:for = "{{iconSize}}">
    < icon type = "success" size = "{{item}}" />
  </block >
  < view class = "title">(2)实现内容变化</view >
  < block wx:for = "{{iconType}}">
    < icon type = "{{item}}" size = "40" />
  </block >
  < view class = "title">(3)实现颜色变化</view >
  < block wx:for = "{{iconColor}}">
```

```
    <icon type="success" size="40" color="{{item}}" />
  </block>
</view>
```

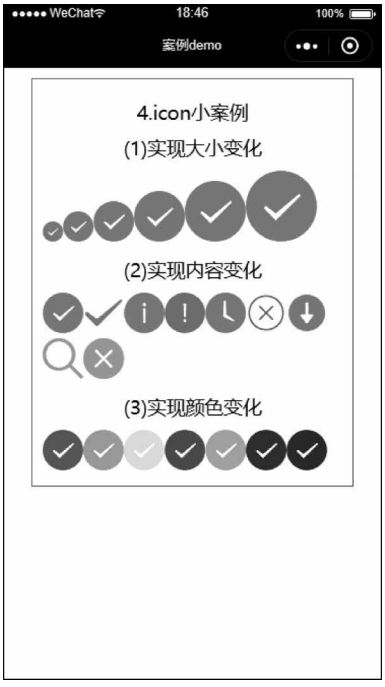


图 3.6　icon 组件小案例

pages/icon/icon.js 文件代码如下：

```
Page({
  data: {
    iconSize: [20, 30, 40, 50, 60, 70],
    iconType: [
      "success", "success_no_circle", "info", "warn", "waiting", "cancel", "download",
"search", "clear"
    ],
    iconColor: [
      "red", "orange", "yellow", "green", "red", "blue", "purple"
    ],
  }
})
```

【代码讲解】　本例在 icon.js 文件中的 data 中设置 3 个数组，分别为 iconSize、iconType 和 iconColor，用于设置图标的大小、图标的类型和图标的颜色。在 icon.wxml 文件中使用< block >标签配合 wx:for 批量生成多个标签组件。

3.3.2　text

< text >为文本组件，用于文字的显示，小程序的文本组件支持转义字符。其属性如

表 3.6 所示。

表 3.6 ＜text＞组件属性

属 性 名	类 型	默 认 值	说 明	最低版本
selectable	boolean	false	文本是否可选	1.1.0
space	string		显示连续空格	1.4.0
decode	boolean	false	是否解码	1.4.0

例 3-5 组件 text 小案例，运行效果如图 3.7 所示。

pages/text/text.wxml 文件代码如下：

视频讲解

```
<view class="demo-box">
  <view class="title">5.text 小案例</view>
  <view class="title">用于文本的显示</view>
  <text>{{text}}</text>
  <button bindtap="add">增加一行</button>
  <button bindtap="reduce">删除一行</button>
</view>
```

(a) 页面初始效果　(b) 增加一行文字　(c) 删除一行文字

图 3.7 text 组件小案例

pages/text/text.js 文件代码如下：

```
var initData = "2019 年,中国要推进这 70 个工程项目：制定实施新时期"互联网 + "行动,实施数字经济、"互联网 + "重大工程,建设人工智能创新应用先导区,持续推进大数据综合试验区建设；加快 5G 商用步伐和 IPv6 规模部署,加强人工智能、工业互联网、物联网等新型基础设施建设和融合应用。"
var extraLine = [];                                    //创建一个空数组
Page({
  data: {
```

```
    text: initData
  },
  add: function(e) {
    extraLine.push("增加一行")                          //增加一行
    this.setData({
      text: initData + '\n' + extraLine.join('\n')      //更新数组值
    })
  },
  reduce: function(e) {
    if(extraLine.length > 0) {
      extraLine.pop()                                   //删除一行
      this.setData({
        text: initData + '\n' + extraLine.join('\n')    //更新数组值
      })
    }
  }
})
```

【代码讲解】 本例在 text.wxml 文件中通过< text >组件存放文字，以及增加两个< button >按钮，分别绑定了点击事件，用于实现增加一行和删除一行的操作，对应在 text.js 文件中自定义 add()和 reduce()两个函数。

图 3.7(a)为初始页面；图 3.7(b)为点击“增加一行”按钮后，在文字内容下方增加一行；图 3.7(c)为点击“删除一行”按钮后，新增加的一行消失，回到页面初始状态。

3.3.3 progress

< progress >为进度条组件，用于进度的显示，长度单位默认为 px。其属性如表 3.7 所示。

表 3.7 ＜progress＞组件属性

属 性 名	类 型	默 认 值	说 明
percent	number		百分比 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/string	6	进度条线的宽度
color	string	#09BB07	进度条颜色(请使用 activeColor)
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：动画从头播；forwards：动画从上次结束点接着播

例如，自定义一个当前进度为 30，宽度为 10rpx 的进度条，示例代码如下：

```
< progress percent = "30" stroke - width = "10rpx"></progress >
```

其运行效果如图 3.8 所示。

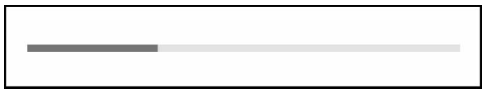


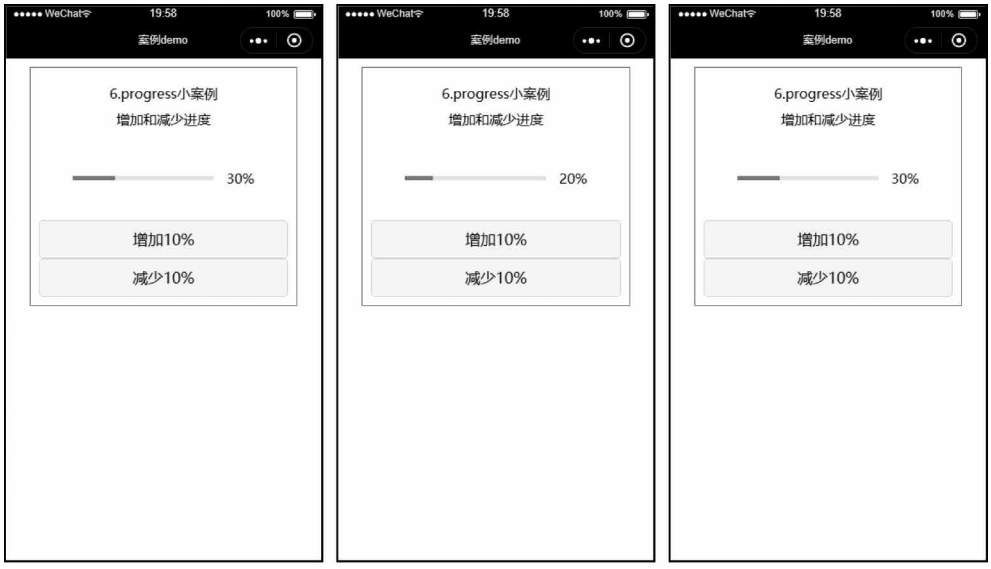
图 3.8　进度条图示

例 3-6　progress 组件小案例，运行效果如图 3.9 所示。

视频讲解

pages/progress/progress.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">6.progress 小案例</view>
  <view class="title">增加和减少进度</view>
  <progress percent="{{progress}}" stroke-width="10rpx" show-info></progress>
  <button bindtap="add">增加 10%</button>
  <button bindtap="reduce">减少 10%</button>
</view>
```



(a) 页面初始效果　　(b) 减少10%　　(c) 增加10%

图 3.9　progress 组件小案例

pages/progress/progress.js 文件代码如下：

```
var per = 30;
Page({
  data: {
    progress: per,                                    //数据初始化
  },
  add: function(e) {
    per += 10;                                        //增加 10%
    if (per > 100) {
      per = 100;                                      //进度超过 100% 不增加
    }
```

```
    this.setData({
      progress: per                                    //更新进度值
    })
  },
  reduce: function(e) {
    per -= 10;                                         //减少 10 %
    if (per < 0) {
      per = 0;                                         //进度小于 0 % 不减少
    }
    this.setData({
      progress: per                                    //更新进度值
    })
  }
})
```

pages/progress/progress.wxss 文件代码如下：

```
progress {
  padding: 80rpx;
}
```

【代码讲解】 本例在 progress.wxml 文件中放置< progress >组件，通过设置属性 show-info 让进度条右边显示当前进度值，以及增加两个< button >按钮，分别绑定了点击事件，用于实现增加 10%的进度值和减少 10%的进度值，在 progress.js 文件中自定义 add()和 reduce()两个函数。

图 3.9(a)为初始页面；图 3.9(b)为点击“减少 10%”按钮后，进度值从 30%变为 20%；图 3.9(c)为点击“增加 10%”按钮后，进度值从 20%变为 30%。

3.4　表单组件

表单组件在网页开发中十分常见，对于小程序而言，当用户需要设计注册、登录等页面时可以使用表单。表单组件是多类组件的统称。

3.4.1　button

< button >为按钮组件，是常用的表单组件之一，用于事件的触发以及表单的提交。其属性如表 3.8 所示。

表 3.8　< button >组件属性

属　性　名	类型	默认值	说　　明	最低版本
size	string	default	按钮的大小	
type	string	default	按钮的样式类型	
plain	boolean	false	按钮是否镂空，背景色透明	
disabled	boolean	false	是否禁用	
loading	boolean	false	名称前是否带 loading 图标	

续表

属　性　名	类型	默认值	说　　明	最低版本
form-type	string		用于<form>组件，点击分别会触发<form>组件的 submit/reset 事件	
open-type	string	button-hover	微信开放能力	1.1.0
hover-class	string	button-hover	指定按钮按下去的样式类。当 hover-class="none"时，没有点击态效果	
hover-stop-propagation	boolean	false	指定是否阻止本节点的祖先节点出现点击态	1.5.0
hover-start-time	number	20	按住后多久出现点击态，单位：ms	
hover-stay-time	number	70	手指松开后点击态保留时间，单位：ms	

注意：button-hover 默认为{background-color：rgba(0，0，0，0.1)；opacity：0.7；}。

size 属性值如下。

(1) default：默认按钮，宽度与手机屏幕宽度保持一致。

(2) mini：迷你按钮，尺寸小于默认按钮。

示例代码如下：

```
<button>默认按钮</button>
<button size="mini">迷你按钮 </button>
```

其运行效果如图 3.10 所示。

type 属性值如下。

(1) primary：主要按钮，颜色为绿色。

(2) default：默认按钮，颜色为灰白色。

(3) warn：警告按钮，颜色为红色。

示例代码如下：

```
<button type="primary">primary 按钮</button>
<button type="default">default 按钮</button>
<button type="warn">warn 按钮</button>
```

其运行效果如图 3.11 所示。

图 3.10　默认按钮和迷你按钮

图 3.11　type 属性值对应的按钮

form-type 属性值如下。

(1) submit：提交表单。

(2) reset：重置表单。

示例代码如下：

```
<button form-type="submit">提交按钮</button>
<button form-type="reset">重置按钮</button>
```

其运行效果如图 3.12 所示。

图 3.12　form-type 属性值对应的按钮

视频讲解

例 3-7　button 组件小案例，运行效果如图 3.13 所示。

图 3.13　button 组件小案例

pages/button/button.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">7.button 小案例</view>
  <view class="title">(1)迷你按钮</view>
  <button size="mini" type="primary">主要按钮</button>
  <button size="mini" type="default">次要按钮</button>
  <button size="mini" type="warn">警告按钮</button>
  <view class="title">(2)按钮状态</view>
  <button>普通按钮</button>
  <button disabled>禁用按钮</button>
  <button loading>加载按钮</button>
  <view class="title">(3)增加按钮事件</view>
  <button bindgetuserinfo="getUserDetail" open-type="getUserInfo">点我获取用户信息
```

```
</button>
</view>
```

pages/button/button.js 文件代码如下：

```
Page({
  getUserDetail: function(e) {
    console.log(e.detail.userInfo)
  }
})
```

pages/button/button.wxss 文件代码如下：

```
button {
  margin: 10rpx;
}
```

【代码讲解】 在 button.wxml 文件中设置 3 组效果，分别为迷你按钮、普通按钮和带点击事件按钮。第 1 组设置相同的 size 属性和不同的 type 属性实现 3 种不同类型的迷你按钮；第 2 组设置属性 disabled 和 loading 实现按钮禁用和加载动画效果；第 3 组通过 bindgetuserinfo="getUserDetail"为按钮增加事件，并追加 open-type="getUserInfo"状态，然后在 JS 文件中定义点击事件。

3.4.2 checkbox

<checkbox>为复选框组件，常用于在表单中进行多项数据的选择。复选框的<checkbox-group>为父控件，其内部嵌套若干个<checkbox>子控件。

<checkbox-group>组件只有一个属性，如表 3.9 所示。

表 3.9 <checkbox-group>组件属性

属性名	类型	默认值	说明
bindchange	eventhandle		<checkbox-group>选中项发生改变时触发 change 事件，detail = {value:[选中的 checkbox 的 value 数组]}

<checkbox>组件的属性如表 3.10 所示。

表 3.10 <checkbox>组件属性

属性名	类型	默认值	说明
value	string		<checkbox>标识，选中时触发<checkbox-group>的 change 事件，并携带<checkbox>的 value
disabled	boolean	false	是否禁用
checked	boolean	false	当前是否选中，可用来设置默认选中
color	color		checkbox 的颜色，同 CSS 的 color

示例代码如下：

```
<checkbox-group>
```

```
  <checkbox value="tiger" checked="true" />老虎
  <checkbox value="elephant" disabled="true" />大象
  <checkbox value="lion" />狮子
  <checkbox value="penguin" />企鹅
</checkbox-group>
```

其运行效果如图 3.14 所示。

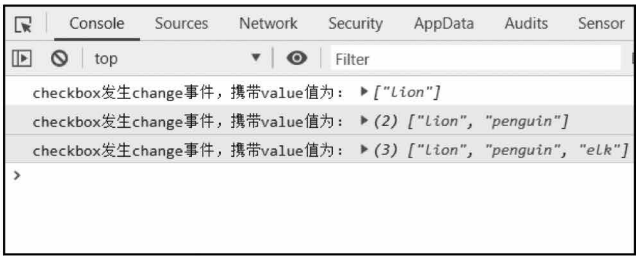
图 3.14　复选框图示

视频讲解

如图 3.14 所示，“老虎”选项被选中，“大象”选项禁止选择，“狮子”和“企鹅”选项均未选中。

例 3-8　checkbox 组件小案例，运行效果如图 3.15 所示。

(a) 多个选项被选中



(b) 多个选项被选中时Console输出的内容

图 3.15　checkbox 组件小案例

pages/checkbox/checkbox.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">8.checkbox 小案例</view>
  <view class="title">利用 for 循环批量生成</view>
  <checkbox-group bindchange="checkboxChange">
    <label wx:for="{{items}}">
      <checkbox value="{{item.name}}" checked="{{item.checked}}" />{{item.value}}
    </label>
  </checkbox-group>
</view>
```

pages/checkbox/checkbox.js 文件代码如下：

```
Page({
  data: {
    items: [
      { name: "tiger", value: "老虎" },
      { name: "elephant", value: "大象" },
      { name: "lion", value: "狮子", checked: "true" },
      { name: "penguin", value: "企鹅" },
      { name: "elk", value: "麋鹿" },
      { name: "swan", value: "天鹅" },
    ]
```

```
  },
  checkboxChange:function(e) {
    console.log("checkbox 发生 change 事件,携带 value 值为: ", e.detail.value)
  }
})
```

【代码讲解】 本例首先在 checkbox.js 文件中的 data 中定义一个数组 items,用于记录复选框的名称(name)、值(value)以及选中情况,并在 check.wxml 文件中使用<checkbox-group>标签包裹<checkbox>,使用<label>标签配合 wx:for 实现批量生成多个 checkbox 组件;其次在<checkbox-group>标签上绑定监听事件,在 checkbox.js 文件中自定义 checkboxChange()函数,以达到每次被触发后都在 Console 控制台输出最新选中的值。

图 3.15(a)为多个选项被选中后的效果;图 3.15(b)为 Console 控制台输出的内容,显示被选中的选项所携带的值。

3.4.3 input

<input>为输入框组件,常用于文本(如姓名、年龄等信息)的输入。其属性如表 3.11 所示。

表 3.11 <input>组件属性

属性名	类型	默认值	说明
value	string		输入框的初始内容
type	string	"text"	input 的类型
password	boolean	false	是否是密码类型
placeholder	string		输入框为空时的占位符
placeholder-style	string		指定 placeholder 的样式
placeholder-class	string		"input-placeholder"指定 placeholder 的样式类
disabled	boolean	false	是否禁用
maxlength	number	140	最大输入长度,设置为-1 的时候不限制最大长度
cursor-spacing	number	0	指定光标与键盘的距离,单位:px(基础库 2.4.0 起支持 rpx)。取 input 距离底部的距离和 cursor-spacing 指定距离的最小值作为光标与键盘的距离
auto-focus	boolean	false	(即将废弃,请直接使用 focus)自动聚焦,拉起键盘
focus	boolean	false	获取焦点
confirm-type	string	"done"	设置键盘右下角按钮的文字,仅在 type='text'时生效
confirm-hold	boolean	false	点击键盘右下角按钮时是否保持键盘不收起
cursor	number		指定 focus 时的光标位置
selection-start	number	-1	光标起始位置,自动聚集时有效,需与 selection-end 搭配使用
selection-end	number	-1	光标结束位置,自动聚集时有效,需与 selection-start 搭配使用
adjust-position	boolean	true	键盘弹起时,是否自动上推页面
bindinput	eventhandle		键盘输入时触发,event.detail = {value, cursor, keyCode},keyCode 为键值,基础库 2.1.0 起支持,处理函数可以直接返回一个字符串,将替换输入框的内容

续表

属　性　名	类　　型	默认值	说　　明
bindfocus	eventhandle		输入框聚焦时触发，event.detail = {value，height}，height为键盘高度，在基础库 1.9.90 起支持
bindblur	eventhandle		输入框失去焦点时触发，event.detail = {value：value}
bindconfirm	eventhandle		点击“完成”按钮时触发，event.detail = {value：value}

type 属性值如下。

(1) text：文本输入键盘。

(2) Number：数字输入键盘。

(3) idcard：身份证输入键盘。

(4) digit：带小数点的数字键盘。

confirm-type 属性值如下。

(1) send：右下角按钮为“发送”。

(2) search：右下角按钮为“搜索”。

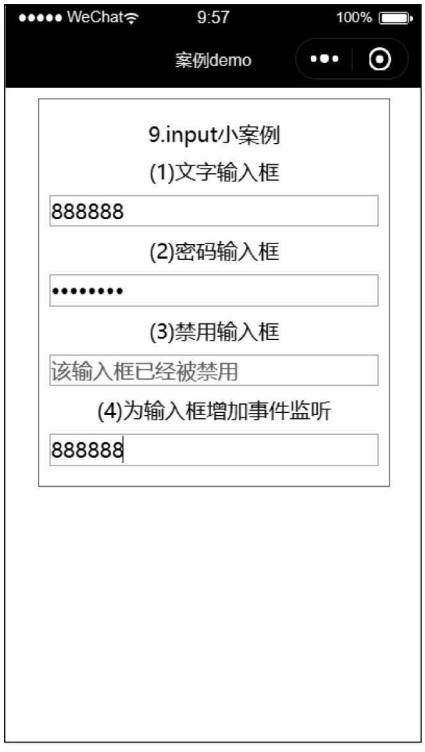
(3) next：右下角按钮为“下一个”。

(4) go：右下角按钮为“前往”。

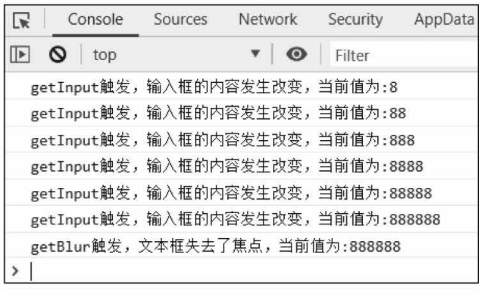
(5) done：右下角按钮为“完成”。

视频讲解

例 3-9　input 组件小案例，运行效果如图 3.16 所示。



(a) 模拟器效果



(b) 控制台效果

图 3.16　input 组件小案例

pages/input/input.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">9.input 小案例</view>
  <view class="title">(1)文字输入框</view>
  <input type="text" maxlength="10" placeholder="这里最多只能输入 10 个字" />
  <view class="title">(2)密码输入框</view>
  <input type="password" placeholder="请输入密码" />
  <view class="title">(3)禁用输入框</view>
  <input disabled placeholder="该输入框已经被禁用" />
  <view class="title">(4)为输入框增加事件监听</view>
  <input bindinput="getInput" bindblur="getBlur" placeholder="这里输入的内容将会被监听"/>
</view>
```

pages/input/input.js 文件代码如下：

```
Page({
  getInput: function(e) {
    console.log("getInput 触发,输入框的内容发生改变,当前值为:" + e.detail.value);
  },
  getBlur: function(e) {
    console.log("getBlur 触发,文本框失去了焦点,当前值为:" + e.detail.value);
  },
})
```

pages/input/input.wxss 文件代码如下：

```
input {
  margin: 20rpx auto; border: 1px solid silver;
}
```

【代码讲解】 本例包含 4 个输入框 input 组件，分别为最大字符长度限时为 10、密码输入框、禁用输入框以及带有监听事件的输入框。在 input.wxml 文件中前三组分别通过设置属性 maxlength、password、disabled 来实现效果；第四组通过属性 bindinput 和 bindfocus 为输入框增加当键盘输入时和失去焦点时所触发的事件，并在 input.js 文件中自定义两个事件函数 getInput()和 getBlur()。

图 3.16(a)是模拟器上的程序效果；图 3.16(b)是输入数据和失去焦点时程序在 Console 控制台打印的结果。

3.4.4 label

<label>是标签组件，label 组件不会呈现任何效果，但是可以用来改进表单组件的可用性。当用户在 label 元素内点击文本时，就会触发此控件，即当用户选择该标签时，事件会传递到和标签相关的表单控件上，可以使用 for 属性绑定 id，也可以将控件放在该标签内部。该组件对应的属性如表 3.12 所示。

表 3.12　＜label＞组件属性

属　性　名	类　　型	说　　明
for	String	绑定控件的 id

注意：目前可以绑定的控件有< button >、< checkbox >、< radio >、< switch >。

这里以复选框< checkbox >为例，使用< label >标签的 for 属性绑定对应复选框，示例代码如下：

```
< checkbox - group >
  < checkbox id = "tiger" value = "tiger" checked = "true" />
  < label for = "tiger">老虎</label >
</checkbox - group >
```

也可以使用< label >包裹< checkbox >标签：

```
< checkbox - group >
  < label >
    < checkbox value = "tiger" checked = "true" />老虎
  </label >
</checkbox - group >
```

视频讲解

上述两种做法效果一样，当用户点击“老虎”内容区域时，< checkbox >组件均被选中。

例 3-10　label 组件小案例，运行效果如图 3.17 所示。

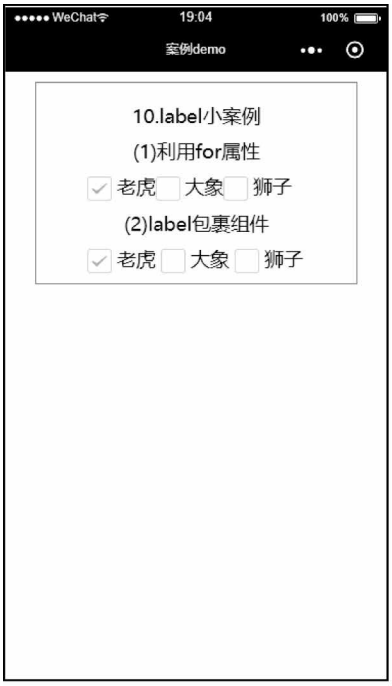
pages/label/label.wxml 文件代码如下：

```
< view class = "demo - box">
  < view class = "title"> 10.label 小案例</view >
  < view class = "title">(1)利用 for 属性</view >
  < checkbox - group >
    < checkbox id = "tiger" checked />
    < label for = "tiger">老虎</label >
    < checkbox id = "elephant" />
    < label for = "elephant">大象</label >
    < checkbox id = "lion" />
    < label for = "lion">狮子</label >
  </checkbox - group >
  < view class = "title">(2)label 包裹组件</view >
  < checkbox - group >
    < label >
      < checkbox checked />老虎
    </label >
    < label >
      < checkbox/>大象
    </label >
    < label >
      < checkbox/>狮子
    </label >
```

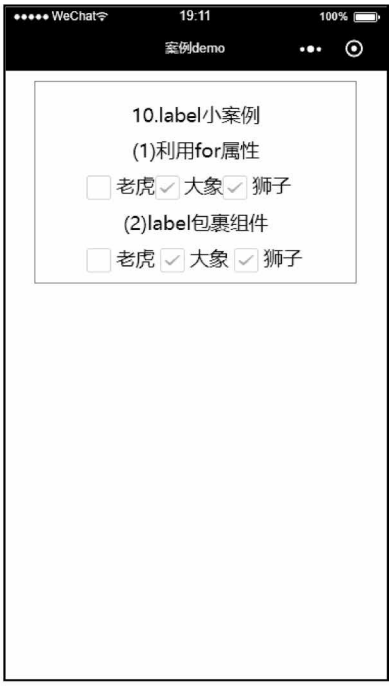
```
  </checkbox-group>
</view>
```

pages/label/label.wxss 文件代码如下：

```
checkbox-group {
  margin: 0 80rpx ;
}
```



(a) “老虎”选项被默认选中



(b) 点击标签后的效果

图 3.17　label 组件小案例

【代码讲解】 本例在 label.wxml 文件中放置两组标签组件，第一组使用 for 属性绑定 id，第二组让 label 包裹<checkbox>组件。

如图 3.17 所示，两种方法效果相同，当分别单击每个文字内容后，如果左边复选框是选中状态，就会变成未选中状态；如果左边复选框是未选中状态，就会变成选中状态。

3.4.5　form

<form>为表单控件组件，用于提交表单组件中的内容。<form>控件组件内部可以嵌套多种组件，具体组件类型如下所示。

(1) <input>：输入框组件；

(2) <button>：按钮组件；

(3) <checkbox>：复选框组件；

(4) <switch>：开关选择器；

（5）＜radio＞：单项框组件；
（6）＜picker＞：滚动选择器；
（7）＜slider＞：滑动选择器；
（8）＜textarea＞：多行输入框；
（9）＜label＞：标签组件。
＜form＞控件组件的属性如表 3.13 所示。

表 3.13　＜form＞控件组件属性

属　性　名	类　　型	说　　明
report-submit	boolean	是否返回 formId 用于发送模板消息
bindsubmit	eventhandle	携带 form 中的数据触发 submit 事件，event.detail = {value : {'name': 'value'} , formId: ''}
bindreset	eventhandle	表单重置时会触发 reset 事件

注意：当表单组件＜form＞需要提交二级内容表单组件（如＜input＞）的内容时，＜form＞组件中需添加 bindsubmit 事件，＜button＞组件需添加 form-type 属性并赋值为 submit，此时的二级内容表单组件（如＜input＞）还需要设置 name 属性。当需要在用户提交表单之后发送模板消息时，表单组件＜form＞需设置 report-submit 属性并赋值为 true。

例 3-11　form 控件组件小案例，运行效果如图 3.18 所示。

视频讲解

(a) 页面初始效果

(b) Console控制台打印的信息

图 3.18　form 组件小案例

pages/form/form.wxml 文件代码如下：

```
<view class = "demo - box">
  <view class = "title"> 11.form 小案例</view >
  <view class = "title">模拟注册功能</view >
  <form bindsubmit = "onSubmit" bindreset = "onReset">
```

```
    <text>用户名:</text>
    <input name="username" type="text" placeholder="请输入你的用户名"></input>
    <text>密码:</text>
    <input name="password" type="password" placeholder="请输入你的密码"></input>
    <text>手机号:</text>
    <input name="phonenumber" type="password" placeholder="请输入你的手机号"></input>
    <text>验证码:</text>
    <input name="code" type="password" placeholder="请输入验证码"></input>
    <button form-type="submit">注册</button>
    <button form-type="reset">重置</button>
  </form>
</view>
```

pages/form/form.js 文件代码如下：

```
Page({
  onSubmit(e) {
    console.log("表单被注册")
    console.log("form 发生了 submit 事件,携带数据为: ", e.detail.value)
  },
  onReset() {
    console.log("form 发生了 reset 事件,表单已被重置")
  }
})
```

pages/form/form.wxss 文件代码如下：

```
input {
  border: 1px solid silver;
}
button {
  margin-top: 20rpx;
}
text {
  font-size: 36rpx;
}
```

【代码讲解】 本示例在 form.wxm 文件中放置一个<form>控件组件，为其绑定监听事件 bindsubmit="onSubmit"和 bindreset="onReset"，用于监听表单的提交和重置。在<form>控件组件内部嵌套 4 个<input>组件，设置属性 type="text"以及 type="password"用于用户名和密码的输入。页面底部放置两个<button>组件，设置属性 form-type="submit"和 form-type="reset"用于提交和重置表单，并在 form.js 文件中自定义 onSubmit()函数和 onReset()函数。

图 3.18(a)为输入数据时模拟器的效果；图 3.18(b)是 Console 控制台打印的注册信息效果。

3.4.6　picker

＜picker＞为滚动选择器，从页面底部弹出供用户选择。根据 mode 属性值的不同共有5种选择器，分别为普通选择器、多列选择器、时间选择器、日期选择器和省市区选择器。

1. 普通选择器

当 mode ＝"selector"时为普通选择器效果，相关属性如表 3.14 所示。

表 3.14　＜picker mode＝"selector"＞组件属性

属　性　名	类　　型	默认值	说　　明
range	Array/ObjectArray	[ ]	mode 为 selector 或 multiSelector 时，range 有效
range-key	string		当 range 是一个 Object Array 时，通过 range-key 来指定 Object 中 key 的值作为选择器显示内容
value	number	0	表示选择了 range 中的第几个(下标从 0 开始)
bindchange	eventhandle		value 改变时触发 change 事件，event.detail ＝ {value: value}
disabled	boolean	false	是否禁用
bindcancel	eventhandle		取消选择或点遮罩层收起 picker 时触发

2. 多列选择器

当 mode＝"multiSelector"时为多列选择器效果，其属性如表 3.15 所示。

表 3.15　＜picker mode＝"multiSelector"＞组件属性

属　性　名	类　　型	默认值	说　　明
range	Array/ObjectArray	[ ]	mode 为 selector 或 multiSelector 时，range 有效
range-key	string		当 range 是一个 Object Array 时，通过 range-key 来指定 Object 中 key 的值作为选择器显示内容
value	number	0	表示选择了 range 中的第几个(下标从 0 开始)
bindchange	eventhandle		value 改变时触发 change 事件，event.detail＝{value: value}
bindcolumnchange	eventhandle		列改变时触发

3. 时间选择器

当 mode＝"time"时为时间选择器效果，其属性如表 3.16 所示。

表 3.16　＜picker mode＝"time"＞组件属性

属　性　名	类　　型	默认值	说　　明
value	string		表示选中的时间，格式为"hh:mm"
start	string		表示有效时间范围的开始，字符串格式为"hh:mm"
end	string		表示有效时间范围的结束，字符串格式为"hh:mm"
bindchange	eventhandle		value 改变时触发 change 事件，event.detail ＝ {value}
bindcolumnchange	eventhandle		列改变时触发

4. 日期选择器

当 mode="date"时为日期选择器效果,其属性如表 3.17 所示。

表 3.17 <picker mode="date">组件属性

属性名	类型	默认值	说明
value	string	0	表示选中的日期,格式为"YYYY-MM-DD"
start	string		表示有效日期范围的开始,字符串格式为"YYYY-MM-DD"
end	string		表示有效日期范围的结束,字符串格式为"YYYY-MM-DD"
fields	string	day	有效值 year,month,day,表示选择器的粒度
bindchange	eventhandle		value 改变时触发 change 事件,event.detail = {value}

5. 省市区选择器

当 mode="region"时为省市区选择器效果,其属性如表 3.18 所示。

表 3.18 <picker mode="region">组件属性

属性名	类型	默认值	说明
value	array	[]	表示选中的省市区,默认选中每一列的第一个值
custom-item	string		可为每一列的顶部添加一个自定义的项
bindchange	eventhandle		value 改变时触发 change 事件,event.detail = {value, code, postcode},其中字段 code 是统计用区划代码,postcode 是邮政编码

例 3-12 picker 组件小案例,运行效果如图 3.19 所示。

视频讲解

pages/picker/picker.wxml 文件代码如下:

```
<view class="demo-box">
  <view class="title">12.表单组件 picker 的简单应用</view>
  <view class="title">(1)普通选择器</view>
  <picker mode="selector" range="{{oneItems}}" bindchange="selectorChange">
    <view>当前选择: {{selector}}</view>
  </picker>
  <view class="title">(2)多列选择器</view>
  <picker mode="multiSelector" range="{{doubleItems}}"
    bindchange="multiSelectorChange">
    <view>当前选择: {{multiSelector}}</view>
  </picker>
  <view class="title">(3)时间选择器</view>
  <picker mode="time" bindchange="timeChange">
    <view>当前选择: {{time}}</view>
  </picker>
  <view class="title">(4)日期选择器</view>
  <picker mode="date" bindchange="dateChange">
    <view>当前选择: {{date}}</view>
  </picker>
  <view class="title">(5)省市区选择器</view>
  <picker mode="region" bindchange="regionChange">
    <view>当前选择: {{region}}</view>
```

(a) 普通选择器　(b) 单击多列选择器　(c) 时间选择器

(d) 日期选择器　(e) 省市区选择器　(f) 选择器确定区域后的效果

图 3.19　picker 组件小案例

```
  </picker>
</view>
```

pages/picker/picker.js 文件代码如下：

```
Page({
  //页面的初始数据
  data: {
    oneItems: ["老虎", "大象", "狮子"],
    doubleItems: [
```

```
      ["千岛湖", "洞庭湖", "玄武湖"],
      ["鼓浪屿", "平遥古城", "坝上草原"],
      ["泰山", "黄山", "华山"]
    ]
  },
  selectorChange: function(e) {
    var i = e.detail.value;                    //获得数组的下标
    var value = this.data.oneItems[i];         //获得选项的值
    this.setData({
      selector: value                          //将用户选择的值更新赋给 selector
    });
  },
  multiSelectorChange: function(e) {
    var arrayIndex = e.detail.value;           //获得数组的下标
    var value = new Array();                   //声明一个空数组,用于存放用户选择的值
    for (var i = 0; i < arrayIndex.length; i++) {
      var m = arrayIndex[i];                   //通过数组的遍历,获得第 i 个数组元素的下标
      var n = this.data.doubleItems[i][m];     //获得第 i 个数组的元素值
      value.push(n);                           //往数组中追加新的值
    }
    this.setData({
      multiSelector: value                     //将用户选择的值更新赋给 multiSelector
    });
  },
  timeChange: function(e) {
    var value = e.detail.value;                //获得选择的时间
    this.setData({
      time: value                              //将用户选择的值更新赋给 time
    });
  },
  dateChange: function(e) {
    var value = e.detail.value;                //获得选择的日期
    this.setData({
      date: value                              //将用户选择的值更新赋给 date
    });
  },
  regionChange: function(e) {
    var value = e.detail.value;                //获得选择的省市区
    this.setData({
      region: value                            //将用户选择的值更新赋给 region
    });
  },
})
```

【代码讲解】 本示例在 picker.wxml 文件中设置了 5 组不同效果的选择器,分别为普通选择器、多列选择器、时间选择器、日期选择器和省市区选择器。在普通选择器中设置属性 mode="selector"并为绑定监听事件,在 JS 文件中自定义选项数组和函数 selectorChange(),在 wxml 文件中通过{{selector}}动态获取用户选择的内容;在多列选择器中设置属性 mode="selector"并为组件绑定监听事件,在 JS 文件中自定义选项数组和函数

multiSelectorChange()，在 WXML 文件中通过{{multiSelector}}动态获取用户选择的内容；在时间选择器中设置属性 mode="time"并为组件绑定监听事件，在 JS 文件中自定义函数 timeChange()，在 WXML 文件中通过{{time}}动态获取用户选择的时间；在日期选择器中设置属性 mode="date"并为组件绑定监听事件，在 JS 文件中自定义函数 dateChange()，在 WXML 文件中通过{{date}}动态获取用户选择的日期；在省市区选择器中设置属性 mode="region"并为组件绑定监听事件，在 JS 文件中自定义函数 regionChange()，在 WXML 文件中通过{{region}}动态获取用户选择的地区。

3.4.7　picker-view

<picker-view>是嵌入页面的滚动选择器，其中只可放置 picker-view-column 组件，其属性如表 3.19 所示。

表 3.19　<picker-view>组件属性

属　性　名	类　　型	默认值	说　　明
value	Array.<number>	[]	数组中的数字依次表示 picker-view 内的 picker-view-column 选择的第几项（下标从 0 开始），数字大于 picker-view-column 可选项长度时，选择最后一项
indicator-style	string		设置选择器中间选中框的样式
indicator-class	string		设置选择器中间选中框的类名
mask-style	string		设置蒙层的样式
mask-class	string		设置蒙层的类名
bindchange	eventhandle		滚动选择时触发 change 事件，event.detail = {value}；value 为数组，表示 picker-view 内的 picker-view-column 当前选择的是第几项（下标从 0 开始）

例 3-13　picker-view 组件小案例，运行效果如图 3.20 所示。

视频讲解

pages/picker-view/picker-view.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">13.picker-view 小案例</view>
  <view class="title">旅游计划</view>
  <view class="title">{{travel}}</view>
  <picker-view value="{{value}}" bindchange="pickerviewChange">
    <picker-view-column>
      <view wx:for="{{touristone}}">{{item}}</view>
    </picker-view-column>
    <picker-view-column>
      <view wx:for="{{touristtwo}}">{{item}}</view>
    </picker-view-column>
    <picker-view-column>
      <view wx:for="{{touristthree}}">{{item}}</view>
    </picker-view-column>
  </picker-view>
</view>
```

pages/picker-view/picker-view.js 文件代码如下：

```
Page({
  //页面的初始数据
  data: {
    touristone: ["五台山", "普陀山", "峨眉山"],
    touristtwo: ["莫高窟", "云冈石窟", "龙门石窟"],
    touristthree: ["法门寺", "佛光寺", "大相国寺"],
    value: [0, 0, 0],                          //设置默认的每个选项的数组下标
  },
  pickerviewChange: function(e) {
    var v = e.detail.value;                    //获得数组的下标
    var travel = [];                           //声明一个空数组,用于存放用户选择的值
    travel.push(this.data.touristone[v[0]]);  //追加用户选择第一个数组的元素
    travel.push(this.data.touristtwo[v[1]]);  //追加用户选择第二个数组的元素
    travel.push(this.data.touristthree[v[2]]);//追加用户选择第三个数组的元素
    this.setData({
      travel: travel                           //将用户选择的值更新赋给 travel
    });
  },
})
```

pages/picker-view/picker-view.wxss 文件代码如下：

```
picker-view {
  width: 100%;
  height: 400rpx;
}
```

(a) 页面初始效果

(b) 更改后的效果

图 3.20　picker-view 组件小案例

【代码讲解】 <picker-view-column>组件是<picker-view>组件的二级组件，本例中picker-view.wxml文件<picker-view>组件包含了3个<picker-view-column>组件，分别用于渲染3个景点。<picker-view>组件和<picker>组件可以达到同样的表现效果，读者可以根据自己的喜好选择使用。

图3.20(a)为页面初始效果；图3.20(b)为用户选择完每个选项后，内容会显示在旅游计划下方。

3.4.8　radio

<radio>为单选框组件，往往需要配合<radio-group>组件来使用，<radio>标签嵌套在<radio-group>当中。<radio-group>组件只有一个属性，如表3.20所示。

表3.20　<radio-group>组件属性

属　性　名	类　　型	说　　明	备　　注
bindchange	eventhandle	<radio-group>选中项发生变化时触发change事件，event.detail = {value:选中项radio的value}	携带值为vent.detail = {value:选中项radio的value}

<radio>组件的属性如表3.21所示。

表3.21　<radio>组件属性

属　性　名	类　　型	默认值	说　　明
value	string		<radio>标识。当该<radio>选中时，<radio-group>的change事件会携带<radio>的value
checked	boolean	false	当前是否选中
disabled	boolean	false	是否禁用
color	color		radio的颜色，同css的color

示例代码如下：

```
<radio-group>
  <radio value="tiger" checked="true">老虎</radio>
  <radio value="elephant" disabled="true">大象</radio>
  <radio value="lion">狮子</radio>
  <radio value="penguin">企鹅</radio>
</radio-group>
```

其效果如图3.21所示。

图3.21　单选框图示

如图3.21所示，“老虎”选项被选中，“大象”选项禁止选择，“狮子”和“企鹅”选项均未选中。<radio-group>组件不允许多选，当前状态有且仅有一个选项被选中，<checkbox-group>组件允许用户多选。

视频讲解

例3-14　radio组件小案例，运行效果如图3.22所示。

pages/radio/radio.wxml文件代码如下：

```
<view class="demo-box">
  <view class="title">14.radio 小案例</view>
  <view class="title">利用 for 循环批量生成</view>
  <radio-group bindchange="radioChange">
    <block wx:for="{{radioItems}}">
      <radio value="{{item.name}}" checked="{{item.checked}}" />{{item.value}}
    </block>
  </radio-group>
</view>
```

(a) “狮子”选项被选中　　(b) 新选项被选中时Console控制台输出的内容

图 3.22　radio 组件小案例

pages/radio/radio.js 文件代码如下：

```
Page({
  data: {
    radioItems: [
      { name: 'tiger', value: '老虎' },
      { name: 'elephant', value: '大象' },
      { name: 'lion', value: '狮子', checked: 'true' },
      { name: 'penguin', value: '企鹅' },
      { name: 'elk', value: '麋鹿' },
      { name: 'swan', value: '天鹅' },
    ]
  },
  radioChange:function(e) {
    console.log("radio 发生 change 事件,携带 value 值为: ", e.detail.value)
  }
```

```
})
```

pages/radio/radio.wxss 文件代码如下：

```
radio-group {
  margin: 0 200rpx;
}
```

【代码讲解】 本例首先在 radio.js 文件中的 data 中定义一个数组 radioitems，用于记录单选框的名称 name、值 value 及选中情况，并在 radio.wxml 文件中使用<radio-group>标签包裹<radio>，使用<label>标签配合 wx：for 实现批量生成多个 radio 组件；其次在<radio-group>标签上绑定监听事件，在 radio.js 文件中自定义 checkboxChange()函数，以达到每次被触发后都在 Console 控制台输出最新选中的所有值。图 3.22(a)为模拟器效果；图 3.22(b)为控制台效果。

3.4.9　slider

<slider>为滑动选择器，用于可视化地动态改变某变量的取值。其属性如表 3.22 所示。

表 3.22　<slider>组件属性

属　性　名	类　　型	默认值	说　　明
min	number	0	最小值
max	number	100	最大值
step	number	1	步长，取值必须大于 0，并且可被(max－min)整除
disabled	boolean	false	是否禁用
value	number	0	当前取值
color	color	#e9e9e9	背景条的颜色(使用 backgroundColor)
selected-color	color	#1aad19	已选择的颜色(使用 activeColor)
backgroundColor	color	#e9e9e9	背景条的颜色
block-size	number	28	滑块的大小，取值范围为 12～28
block-color	color	#ffffff	滑块的颜色
show-value	boolean	false	是否显示当前 value
bindchange	eventhandle		完成一次拖动后触发的事件，event.detail = {value: value}
bindchanging	eventhandle		拖动过程中触发的事件，event.detail = {value: value}

例如，设置一个自定义滑动条，最小值为 10、最大值为 100，在右侧显示当前数值，示例代码如下：

```
<slider min="10" max="100" show-value />
```

其运行效果如图 3.23 所示。

图 3.23　滑动条图示

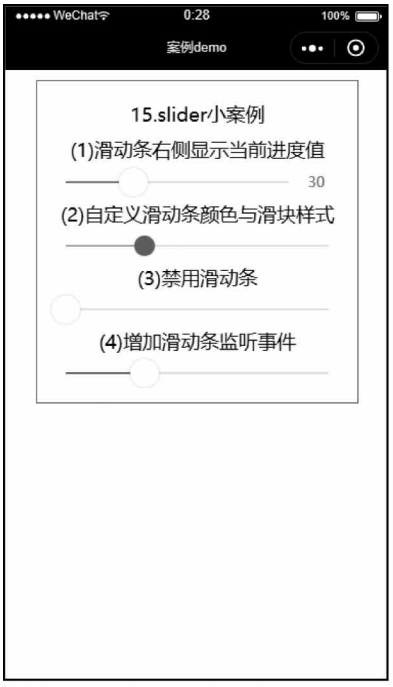
滑动条上的滑块向右边滑动时，右侧的数值会逐渐增大。

视频讲解

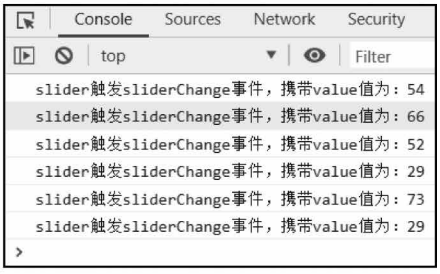
例 3-15　slider 组件小案例，运行效果如图 3.24 所示。

pages/slider/slider.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">14.slider 小案例</view>
  <view class="title">(1)滑动条右侧显示当前进度值</view>
  <slider min="0" max="100" value="30" step="10" show-value="true" />
  <view class="title">(2)自定义滑动条颜色与滑块样式</view>
  <slider min="0" max="100" value="30" block-size="20" block-color="gray"
activeColor="skyblue" />
  <view class="title">(3)禁用滑动条</view>
  <slider min="0" max="100" value="30" disabled="true" />
  <view class="title">(4)增加滑动条监听事件</view>
  <slider min="0" max="100" value="30" bindchange="sliderChange" />
</view>
```



(a) 页面初始效果



(b) Console控制台打印的slider变化

图 3.24　slider 组件小案例

pages/slider/slider.js 文件代码如下：

```
Page({
  sliderChange(e) {
    console.log("slider 触发 sliderChange 事件，携带 value 值为：" + e.detail.value)
  }
})
```

【代码讲解】 本例在 slider.wxml 文件中设置 4 种情况，分别为滑动条右侧显示当前进度值、自定义滑动条颜色和滑块样式、禁用滑动条(无法改变当前数值)、增加滑动条监听事件。

图 3.24(a)是页面初始状态，第 4 个滑动条为其滑块增加监听事件并在 slider.js 文件中自定义函数；图 3.24(b)所示，当第 4 个滑动条被拖动时，会在 Console 控制台上输出 slider 的最新值。

3.4.10　switch

＜switch＞为开关选择器，常用于表单上的开关功能，其属性如表 3.23 所示。

表 3.23　＜switch＞组件属性

属　性　名	类　　型	默认值	说　　明
checked	boolean	false	是否选中
disabled	boolean	false	是否禁用
type	string	switch	样式，有效值：switch，checkbox
bindchange	eventhandle		value 改变时触发 change 事件，event.detail = {value：value}
color	color		switch 的颜色，同 CSS 的 color

示例代码如下：

```
<switch checked = "true" />选中
<switch/>未选中
```

其运行效果如图 3.25 所示。

图 3.25　开关图示

当按钮在右边时为选中状态，当按钮在左边时为未选中状态。

例 3-16　switch 组件小案例，运行效果如图 3.26 所示。

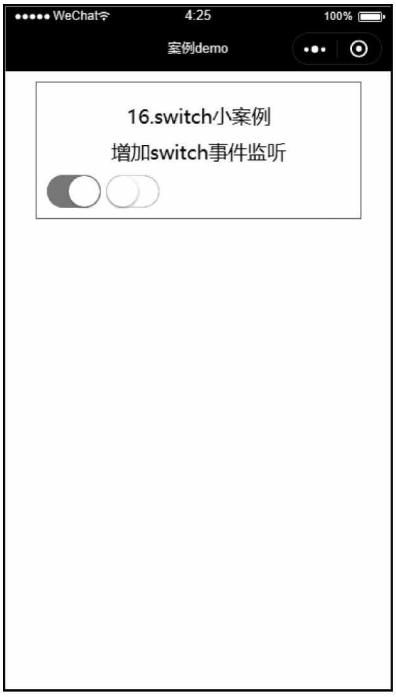
视频讲解

pages/switch/switch.wxml 文件代码如下：

```
<view class = "demo - box">
  <view class = "title"> 16.swtich 小案例</view>
  <view class = "title">增加 switch 事件监听</view>
  <switch checked bindchange = "switch1Change"></switch>
  <switch bindchange = "switch2Change"></switch>
</view>
```

pages/switch/switch.js 文件代码如下：

```
Page({
  switch1Change1: function(e) {
    console.log("switch1 触发 switch1Change 事件，携带 value 值为: " + e.detail.value)
  },
  switch2Change2: function(e) {
    console.log("switch2 触发 switch2Change 事件，携带 value 值为: " + e.detail.value)
  }
})
```



(a) 页面初始效果

(b) Console控制台输出的switch状态变化

图 3.26　switch 组件小案例

【代码讲解】 本例在 switch.wxml 文件中放置两组< switch >标签，第 1 组设置属性 checked 实现选中状态，第 2 组默认为未选中状态，并为两个组件分别绑定监听事件；在 switch.js 文件中自定义 switch1Change()函数和 switch2Change()函数，当 switch 组件被触发后会在 Console 控制台输出当前状态。

图 3.26(a)为页面初始效果图；图 3.26(b)为 Console 控制台输出当前两组 switch 的选中状态。

3.4.11　textarea

< textarea >为多行输入框，常用于多行文字的输入。其属性如表 3.24 所示。

表 3.24　＜textarea＞组件属性

属　性　名	类　　型	默　认　值	说　　明
value	string		输入框的内容
placeholder	string		输入框为空时占位符
placeholder-style	string	switch	指定 placeholder 的样式，目前仅支持 color、font-size 和 font-weight
placeholder-class	string	textarea-placeholder	指定 placeholder 的样式类
disabled	boolean	false	是否禁用
maxlength	number	140	最大输入长度，设置为－1 的时候不限制最大长度

续表

属　性　名	类　　型	默　认　值	说　　明
auto-focus	boolean	false	自动聚焦，拉起键盘
focus	boolean	false	获取焦点
auto-height	boolean	false	是否自动增高，设置 auto-height 时，style.height 不生效
fixed	boolean	false	如果 textarea 是在一个 position:fixed 的区域，需要显示指定属性 fixed 为 true
cursor-spacing	number/string	0	指定光标与键盘的距离，单位：px（基础库 2.4.0 起支持）。取 textarea 底部的距离和 cursor-spacing 指定的距离的最小值作为光标与键盘的距离
cursor	number		指定 focus 时的光标位置（基础库 1.5.0 起支持）
show-confirm-bar	boolean	true	是否显示键盘上方带有“完成”按钮那一栏（基础库 1.6.0 起支持）
selection-start	number	−1	光标起始位置，自动聚集时有效，需与 selection-end 搭配使用（基础库 1.9.0 起支持）
selection-end	number	−1	光标结束位置，自动聚集时有效，需与 selection-start 搭配使用（基础库 1.9.0 起支持）
adjust-position	boolean	true	键盘弹起时，是否自动上推页面（基础库 1.9.0 起支持）
bindfocus	eventhandle		输入框聚焦时触发，event.detail = {value, height}，height 为键盘高度（基础库 1.9.0 起支持）
bindblur	eventhandle		输入框失去焦点时触发，event.detail = {value, cursor}
bindlinechange	eventhandle		输入框行数变化时调用，event.detail = {height: 0 heightRpx: 0, lineCount: 0}
bindinput	eventhandle		当键盘输入时，触发 input 事件，event.detail = {value, cursor, keyCode}，keyCode 为键值，目前工具还不支持返回 keyCode 参数。bindinput 处理函数的返回值并不会反映到 textarea 上
bindconfirm	eventhandle		点击完成时，触发 confirm 事件，event.detail = {value: value}

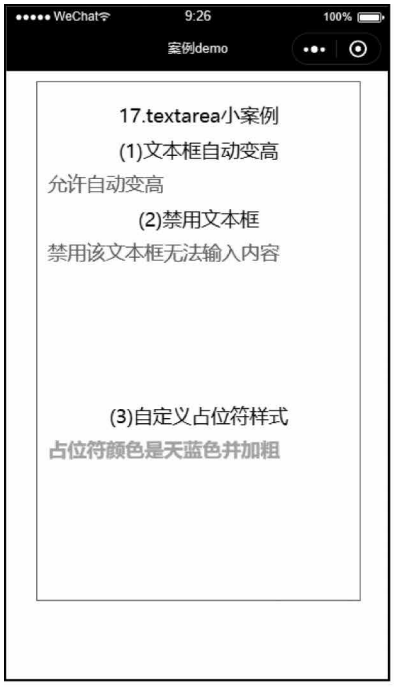
例 3-17　textarea 组件小案例，运行效果如图 3.27 所示。

视频讲解

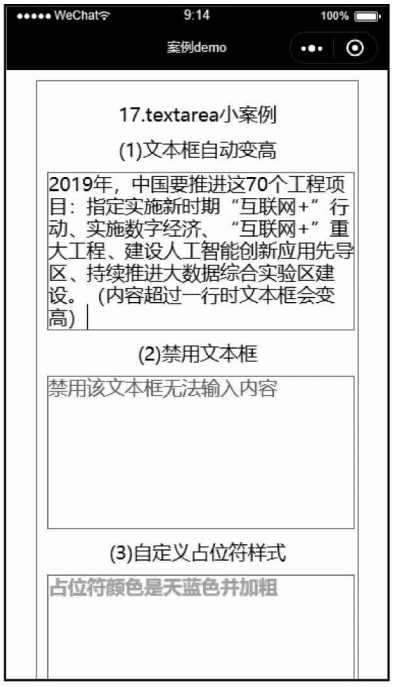
pages/textarea/textarea.wxml 文件代码如下：

```
<view class = "demo - box">
  <view class = "title"> 17.textarea 小案例</view >
  <view class = "title">(1)文本框自动变高</view >
```

```
    <textarea auto-height placeholder="允许自动变高" />
    <view class="title">(2)禁用文本框</view>
    <textarea placeholder="禁用该文本框无法输入内容" disabled/>
    <view class="title">(3)自定义占位符样式</view>
    <textarea placeholder="占位符颜色是天蓝色并加粗" placeholder-style="color:skyblue;
font-weight:bold" />
</view>
```



(a) 页面初始效果



(b) 第一组文本框在输入的内容超过一行时自动变高

图 3.27　textarea 组件小案例

【代码讲解】 本例在 switch.wxml 文件中放置 3 组<textarea>标签：第 1 组通过设置属性 auto-height，允许文本框在输入内容超过一行时自动变高；第 2 组通过设置属性 disabled，文本框被禁用无法进行输入；第 3 组通过设置属性“color:skyblue;font-weight:bold”，使文本框中的占位符呈现天蓝色并加粗。

图 3.27(a)为页面初始效果；图 3.27(b)中第 1 组文本框在输入的内容超过一行时，文本框高度会自动发生变化，第 2 组文本框被禁，第 3 组文本框的占位符样式发生改变。

3.5　导航组件

<navigator>导航组件用于页面之间的跳转，是使用频繁的组件之一。其属性如表 3.25 所示。

表 3.25　＜navigator＞组件属性

属　性　名	类　　型	默认值	说　　明
target	string	self	是在哪个目标上发生跳转，默认为当前小程序
url	string	false	当前小程序内的跳转链接
open-type	navigate	switch	跳转方式

其中，open-type 属性对应 6 种取值。其属性如表 3.26 所示。

表 3.26　open-type 属性

属　性　名	说　　明
navigate	对应 wx.navigateTo 或 wx.navigateToMiniProgram 的功能
redirect	对应 wx.redirectTo 的功能
switchTab	navigate 跳转方式
reLaunch	对应 wx.reLaunch 的功能
navigateBack	对应 wx.navigateBack 的功能
exit	退出小程序，target＝"miniProgram"时生效

例 3-18　navigator 组件小案例，运行效果如图 3.28 所示。

视频讲解

pages/navigator/navigator.wxml 文件代码如下：

```
<view class = "demo - box">
  <view class = "title"> 18.导航组建 navigator 小案例</view >
  <view class = "title">(1)点击打开新页面</view >
  <navigator url = "../new/new" open - type = "navigate">
    <button type = "primary">点击按钮跳转到新页面</button >
  </navigator >
  <view class = "title">(2)点击重定向到新页面</view >
  <navigator url = "../redirect/redirect" open - type = "redirect">
    <button type = "primary">点击按钮跳转到当前页面</button >
  </navigator >
</view >
```

在 app.js 文件中通过 pages/new/new、pages/redirect/redirect 注册两个新页面。

pages/new/new.wxml 文件代码如下：

```
<text >这里是新窗口打开的新页面，点击左上角图标可以返回上一页</text >
```

pages/redirect/redirect.wxml 文件中代码如下：

```
<text >重定向的新页面，无法返回到 navigator.wxml 页面</text >
```

【代码讲解】　本例创建 3 个页面：navigator 页面是项目首页；new 和 redirect 页面是 navigator 页面中两种不同的跳转方式的着陆页。

图 3.28(a)为页面初始效果；图 3.28(b)页面利用 open-type＝"navigate"方式跳转而来，着陆页左上角的返回图标可以返回上一页；图 3.28(c)页面由 open-type＝"redirect"方式跳转而来，着陆页左上角没有返回图标。

(a) 页面初始效果　(b) 点击第1个按钮跳转到新页面　(c) 点击第2个按钮在当前页打开

图 3.28　navigator 组件小案例

3.6　媒体组件

媒体组件用于播放媒体内容，常用的有音频组件、图片组件和视频组件。

3.6.1　audio

＜audio＞为音频组件，用于网络音频的播放。其属性如表 3.27 所示。

表 3.27　＜audio＞组件属性

属　性　名	类　　型	默认值	说　　明
id	string		audio 组件的唯一标识符
src	string	false	要播放音频的资源地址
loop	boolean	false	是否循环播放
controls	boolean	false	是否显示默认控件
poster	string		默认控件上的音频封面的图片资源地址，如果 controls 属性值为 false，则设置 poster 无效
name	string	未知音频	默认控件上的音频名字，如果 controls 属性值为 false，则设置 name 无效
author	string	未知作者	默认控件上的作者名字，如果 controls 属性值为 false，则设置 author 无效
binderror	eventhandle		当发生错误时触发 error 事件，detail = {errMsg: MediaError. code}

续表

属　性　名	类　　型	默认值	说　　明
bindplay	eventhandle		当开始/继续播放时触发 play 事件
bindpause	eventhandle		当暂停播放时触发 pause 事件
bindtimeupdate	eventhandle		当播放进度改变时触发 imeupdate 事件，detail = {currentTime，duration}
bindended	eventhandle		当播放到末尾时触发 ended 事件

其中，binderror 属性触发后的返回值 MediaError. code 共有 4 种取值：

（1）获取资源被用户禁止；

（2）网络错误；

（3）解码错误；

（4）不合适资源。

视频讲解

例 3-19　audio 组件小案例，运行效果如图 3.29 所示。

(a) 页面初始效果　　(b) 音频正在播放　　(c) 音频回到10秒处

图 3.29　audio 组件小案例

pages/audio/audio. wxml 文件代码如下：

```
<view class = "demo - box">
  <view class = "title">19.audio 小案例</view>
  <view class = "title">网络音频播放</view>
  <view>
    <audio poster = "{{poster}}" name = "{{name}}" author = "{{author}}"
  src = "{{src}}" id = "myAudio" controls loop>
    </audio>
    <button size = "mini" bindtap = "audioPlay">播放</button>
```

```
    <button size = "mini" bindtap = "audioPause">暂停</button>
    <button size = "mini" bindtap = "audio10">设置当前播放时间为 10 秒</button>
  </view>
</view>
```

pages/audio/audio.js 文件代码如下：

```
Page({
  onReady: function(e) {
    //使用 wx.createAudioContext 获取 audio 上下文 context
    this.audioCtx = wx.createAudioContext('myAudio')
  },
  data: {
    poster: 'http://y.gtimg.cn/music/photo_new/T002R300x300M00000
             3rsKF44GyaSk.jpg?max_age = 2592000',
    name: '此时此刻',
    author: '许巍',
    src: 'http://ws.stream.qqmusic.qq.com/M500001VfvsJ21xFqb.mp3?guid =
         ffffffff82def4af4b12b3cd9337d5e7&uin = 346897220&vkey = 6292F51E1E384
         E06DCBDC9AB7C49FD713D632D313AC4858BACB8DDD29067D3C601481D36E62053
         BF8DFEAF74C0A5CCFADD6471160CAF3E6A&fromtag = 46',
  },
  audioPlay: function() {
    this.audioCtx.play()                    //播放音乐
  },
  audioPause: function() {
    this.audioCtx.pause()                   //暂停播放
  },
  audio10: function() {
    this.audioCtx.seek(10)                  //回到 10s
  },
})
```

【代码讲解】 本例先在 audio.wxml 文件中放置<audio>组件，然后设置 3 组<button>按钮，分别用于控制音频的播放、暂停以及定位到 10s 处的播放位置。<audio>组件的属性值在 audio.js 文件中定义，包括音频封面图片、歌曲名、演唱者和音频来源，并为 3 组按钮绑定点击事件，在 audio.js 文件中自定义 audioPlay()函数、audioPause()函数和 audio10()函数。

图 3.29(a)为页面初始状态；图 3.29(b)为点击“播放”按钮后，网络音频呈现播放状态，音频处于 0 分 23 秒；图 3.29(c)为网络音频正在播放时，点击“设置当前播放时间为 10 秒”按钮后，音频回到 0 分 10 秒处播放。

3.6.2 image

<image>为图片组件，常用于浏览图片，通过设置属性 mode 能够对图片进行裁剪和缩放，共有 13 种模式，其中 4 种是缩放模式，9 种是裁剪模式，其属性如表 3.28 所示。

表 3.28　＜image＞组件属性

属　性　名	类　　型	默认值	说　　明
src	string		图片资源地址，支持云文件 ID(基础库 2.2.3 起支持)
mode	string	'scaleToFill'	图片裁剪、缩放的模式
lazy-load	boolean	false	图片懒加载，在即将进入当前屏幕可视区域时才开始加载(基础库 1.5.0 起支持)
binderror	eventhandle		当错误发生时，发布到 AppService 的事件名，事件对象 event.detail = {errMsg: 'something wrong'}
bindload	eventhandle		当图片载入完毕时，发布到 AppService 的事件名，事件对象 event.detail = {height:'图片高度 px', width:'图片宽度 px'}

image 组件默认宽度 300px、高度 225px。image 组件中二维码/小程序码图片不支持长按识别，仅在 wx.previewImage()中支持长按识别。

mode 有效值的属性名及说明如表 3.29 所示。

表 3.29　mode 有效值的属性名及说明

属　性　名		说　　明
缩放模式	scaleToFill	不保持纵横比例缩放图片，使图片的宽、高完全拉伸至填满 image 元素
	aspectFit	保持纵横比例缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来
	aspectFill	保持纵横比例缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取
	widthFix	宽度不变，高度自动变化，保持原图宽、高比不变
裁剪模式	top	不缩放图片，只显示图片的顶部区域
	bottom	不缩放图片，只显示图片的底部区域
	center	不缩放图片，只显示图片的中间区域
	left	不缩放图片，只显示图片的左边区域
	right	不缩放图片，只显示图片的右边区域
	top left	不缩放图片，只显示图片的左上边区域
	top right	不缩放图片，只显示图片的右上边区域
	bottom left	不缩放图片，只显示图片的左下边区域
	bottom right	不缩放图片，只显示图片的右下边区域

例 3-20　image 组件小案例。


视频讲解

pages/image/image.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">20.audio 小案例</view>
  <view class="title">(1)不保持纵横比缩放图片，使图片完全适应</view>
  <image src="{{src}}" mode="scaleToFill"></image>
  <view class="title">(2)保持纵横比缩放图片，使图片的长边能完全显示出来</view>
  <image src="{{src}}" mode="aspectFit"></image>
  <view class="title">(3)保持纵横比缩放图片，只保证图片的短边能完全显示出来</view>
  <image src="{{src}}" mode="aspectFill"></image>
  <view class="title">(4)宽度不变，高度自动变化，保持原图宽高比不变</view>
```

```
  <image src="{{src}}" mode="widthFix"></image>
  <view class="title">(5)不缩放图片,只显示图片的顶部区域</view>
  <image src="{{src}}" mode="top"></image>
  <view class="title">(6)不缩放图片,只显示图片的底部区域</view>
  <image src="{{src}}" mode="bottom"></image>
  <view class="title">(7)不缩放图片,只显示图片的中间区域</view>
  <image src="{{src}}" mode="center"></image>
  <view class="title">(8)不缩放图片,只显示图片的左边区域</view>
  <image src="{{src}}" mode="left"></image>
  <view class="title">(9)不缩放图片,只显示图片的右边区域</view>
  <image src="{{src}}" mode="right"></image>
  <view class="title">(10)不缩放图片,只显示图片的左上边区域</view>
  <image src="{{src}}" mode="top left"></image>
  <view class="title">(11)不缩放图片,只显示图片的右上边区域</view>
  <image src="{{src}}" mode="top right"></image>
  <view class="title">(12)不缩放图片,只显示图片的左下边区域</view>
  <image src="{{src}}" mode="bottom left"></image>
  <view class="title">(13)不缩放图片,只显示图片的右下边区域</view>
  <image src="{{src}}" mode="bottom right"></image>
</view>
```

pages/image/image.wxss 文件代码如下：

```
image {
  width: 300rpx;
  height: 300rpx;
  margin-left: 150rpx;
}
```

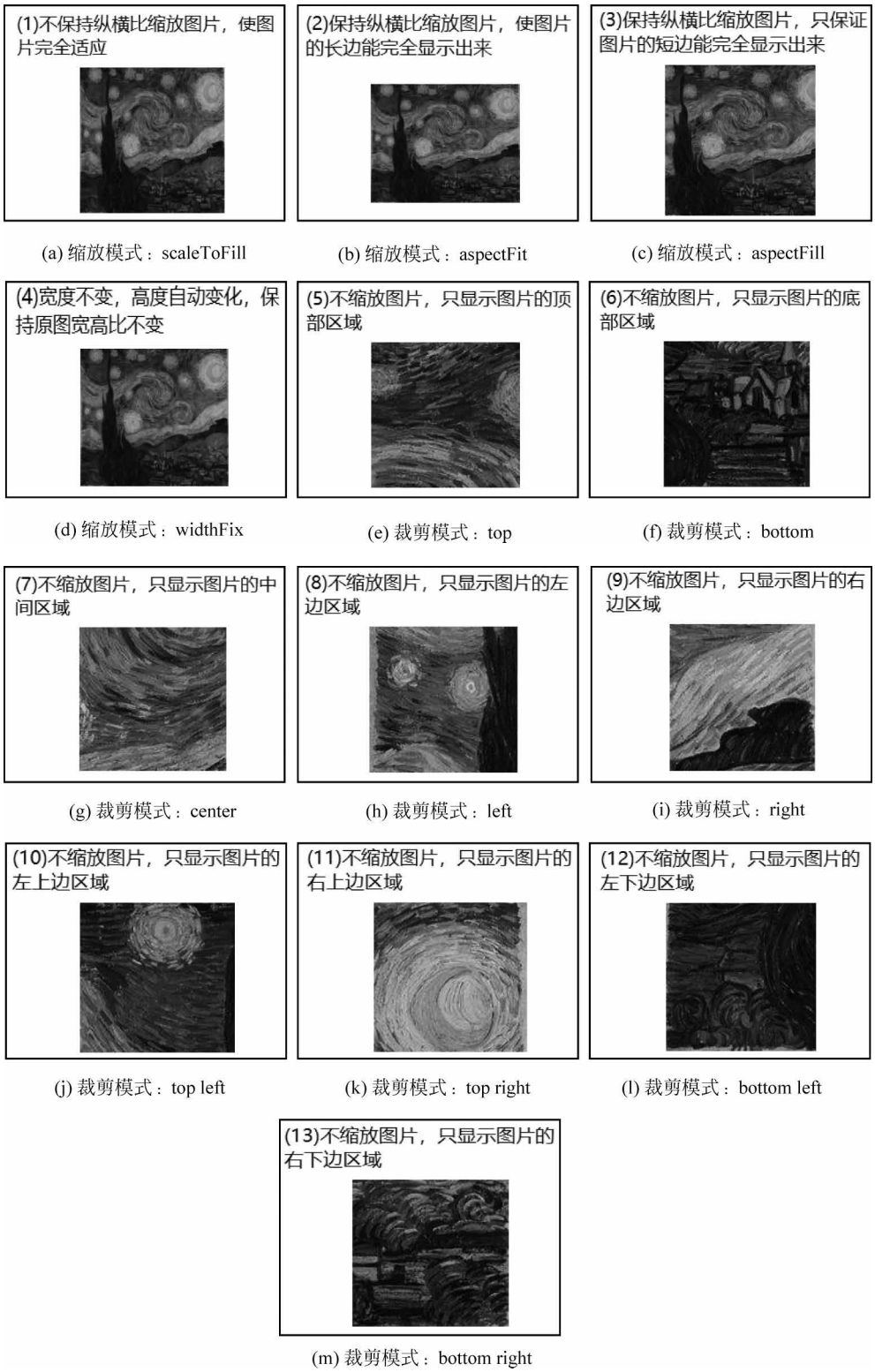
pages/image/image.js 文件代码如下：

```
Page({
  data: {
    src: "/images/xingkong.jpg"
  }
})
```

本例在 images 文件夹下存放素材图片 xingkong.jpg，图片选择梵高的名画《星空》，原图如图 3.30 所示。

图 3.30　图片素材

运行效果如图 3.31 所示。



(a) 缩放模式：scaleToFill　(b) 缩放模式：aspectFit　(c) 缩放模式：aspectFill

(d) 缩放模式：widthFix　(e) 裁剪模式：top　(f) 裁剪模式：bottom

(g) 裁剪模式：center　(h) 裁剪模式：left　(i) 裁剪模式：right

(j) 裁剪模式：top left　(k) 裁剪模式：top right　(l) 裁剪模式：bottom left

(m) 裁剪模式：bottom right

图 3.31　image 组件小案例

【代码讲解】 本例在 image.wxml 文件中放置了 13 组<image>组件，通过设置不同的 mode 属性值，实现对图片的缩放和剪裁。

3.6.3 video

<video>为视频组件，用于播放本地或网络的视频资源，视频的默认宽度为 300px、高度为 225px，视频组件提供弹幕功能，其属性如表 3.30 所示。

表 3.30 <video>组件属性

属性名	类型	默认值	说明
src	string		要播放视频的资源地址，支持云文件 ID(2.3.0)
duration	number		指定视频时长
controls	boolean	true	是否显示默认播放控件(播放/暂停按钮、播放进度、时间)
danmu-list	Array.<object>		弹幕列表
enable-danmu	boolean	false	是否展示弹幕，只在初始化时有效，不能动态变更
autoplay	boolean	false	是否自动播放
loop	boolean	false	是否循环播放
muted	boolean	false	是否静音播放
initial-time	number		指定视频初始播放位置
page-gesture	boolean	false	在非全屏模式下，是否开启亮度与音量调节手势
direction	number		设置全屏时视频的方向，不指定则根据宽、高比自动判断
show-progress	boolean	true	若不设置，宽度大于 240 时才会显示
show-fullscreen-btn	boolean	true	是否显示全屏按钮
show-play-btn	boolean	true	是否显示视频底部控制栏的播放按钮
show-center-play-btn	boolean	true	是否显示视频中间的播放按钮
enable-progress-gesture	boolean	true	是否开启控制进度的手势
object-fit	string	contain	当视频大小与 video 容器大小不一致时，视频的表现形式
poster	string		视频封面的图片网络资源地址或云文件 ID。若 controls 属性值为 false 则设置 poster 无效
show-mute-btn	boolean	false	是否显示静音按钮
title	string		视频的标题，全屏时在顶部展示
play-btn-position	string	bottom	播放按钮的位置
enable-play-gesture	boolean	false	是否开启播放手势，即双击切换播放/暂停
auto-pause-if-navigate	boolean	true	当跳转到其他小程序页面时，是否自动暂停本页面的视频

续表

属　性　名	类　　型	默认值	说　　明
auto-pause-if-open-native	boolean	true	当跳转到其他微信原生页面时，是否自动暂停本页面的视频
vslide-gesture	boolean	false	在非全屏模式下，是否开启亮度与音量调节手势(同 page-gesture)
vslide-gesture-in-fullscreen	boolean	true	在全屏模式下，是否开启亮度与音量调节手势
bindplay	eventhandle		当开始/继续播放时触发 play 事件
bindpause	eventhandle		当暂停播放时触发 pause 事件
bindended	eventhandle		当播放到末尾时触发 ended 事件
bindtimeupdate	eventhandle		播放进度变化时触发，event.detail = {currentTime，duration}。触发频率 250ms 一次
bindfullscreenchange	eventhandle		视频进入和退出全屏时触发，event.detail = {fullScreen，direction}，direction 有效值为 vertical 或 horizontal
bindwaiting	eventhandle		视频出现缓冲时触发
binderror	eventhandle		视频播放出错时触发
bindprogress	eventhandle		加载进度变化时触发，只支持一段加载。event.detail = {buffered}，百分比

例 3-21　video 组件小案例，运行效果如图 3.32 所示。

视频讲解

pages/video/video.wxml 文件代码如下：

```
<view class="demo-box">
  <view class="title">21.video 小案例</view>
  <video id="myVideo" src="{{src}}" danmu-list="{{danmuList}}"
    enable-danmu controls></video>
  <view class="weui-label">弹幕内容:</view>
  <input bindblur="bindInputBlur" type="text" placeholder="在此处输入弹幕内容" />
  <button size="mini" bindtap="bindPlay">播放</button>
  <button size="mini" bindtap="bindPause">暂停</button>
  <button bindtap="bindSendDanmu" size="mini" formType="submit">发送弹幕</button>
</view>
```

pages/video/video.js 文件代码如下：

```
Page({
  onReady: function(res) {
    this.videoContext = wx.createVideoContext('myVideo')
  },
  inputValue: '',                                    //创建一个空字符串用于保存弹幕内容
  data: {
    src: "http://wxsnsdy.tc.qq.com/105/20210/snsdyvideodownload?filekey=
30280201010421301f0201690402534804102ca905ce620b1241b726bc41dcff44e002040128
82540400&bizid=1023&hy=SH&fileparam=302c020101042530230204136ffd93020457e3c4
```

```
ff02024ef202031e8d7f02030f42400204045a320a0201000400",
    danmuList: [{
        text: "第 1s 出现的弹幕",
        color: "#ff0000",
        time: 1
      },
      {
        text: "第 2s 出现的弹幕",
        color: "#ff00ff",
        time: 2
      }
    ]
  },
  bindInputBlur: function(e) {
    this.inputValue = e.detail.value          //获取输入内容
  },
  bindSendDanmu: function() {
    this.videoContext.sendDanmu({
      text: this.inputValue,                  //发送弹幕
    })
  },
  bindPlay: function() {
    this.videoContext.play()                  //播放视频
  },
  bindPause: function() {
    this.videoContext.pause()                 //暂停视频
  },
})
```

(a) 页面初始状态　　(b) 视频播放状态　　(c) 发送弹幕

图 3.32　video 组件小案例

pages/video/video.wxss 文件代码如下：

```
input {
  border: 1px solid gray;
}
```

【代码讲解】 本例在 video.wxml 文件中放置< video >组件并设置其相关属性，< input >组件用来输入弹幕内容，通过 3 组< button >组件分别绑定点击事件 bindPlay()、bindPause()、bindSendDanmu()来实现视频播放、暂停和发送弹幕，并在 video.js 文件中定义视频播放的资源地址及自定义事件函数。

图 3.32(a)为初始化界面，呈现暂停状态；图 3.32(b)为点击“播放”按钮后，视频呈现播放状态；图 3.32(c)为在输入框输入内容“12345678”并点击“发送弹幕”按钮后，输入框中的内容会呈现在屏幕上。

3.7　地图组件

< map >为地图组件，用于地图的展示。小程序使用的地图来自腾讯地图，地图组件为用户提供视角中心点地位、缩放层级的设置、标记物的增加以及内部组件的事件绑定，其属性如表 3.31 所示。

表 3.31　< map >组件属性

属　性　名	类　　型	默认值	说　　明
longitude	number		中心经度
latitude	number		中心纬度
scale	number	16	缩放级别，取值范围为 3～20
marker	Array.< marker >		弹幕列表(基础库 1.0.0 起支持)
cover	Array.< cover >		即将移除，请使用 markers
polyline	Array.< polyline >		路线
circle	Array.< circles >		圆
control	Array.< control >	false	控件(即将废弃，建议使用 cover-view 代替)
include-point	Array.< point >		缩放视野以包含所有给定的坐标点
show-location	boolean	false	显示带有方向的当前定位点
bindtap	eventhandle		点击地图时触发
bindmarkertap	eventhandle		点击标记点时触发，会返回 marker 的 id
bindcontroltap	eventhandle		点击控件时触发，会返回 control 的 id
bindcallouttap	eventhandle		点击标记点对应的气泡时触发，会返回 marker 的 id
bindupdated	eventhandle		视野发生变化时触发
bindregionchange	eventhandle		是否开启控制进度的手势(基础库 1.9.0 起支持)
bindpoitap	eventhandle		点击地图 poi 点时触发

其中，marker 为标记点，用于在地图上显示标记的位置，其属性如表 3.32 所示。

表 3.32　marker 属性

属性名	说　明	类　型	备　注
id	标记点 id	number	marker 点击事件回调会返回此 id。建议为每个 marker 设置 number 类型 id，保证更新 marker 时有更好的性能
latitude	纬度	number	浮点数，范围为－90°～90°
longitude	经度	number	浮点数，范围为－180°～180°
title	标注点名	string	value 改变时触发 change 事件，event.detail = {value：value}
iconPath	显示的图标	string	项目目录下的图片路径，支持相对路径写法，以'/'开头则表示相对小程序根目录；也支持临时路径和网络图片
rotate	旋转角度	number	顺时针旋转的角度，范围为 0°～360°，默认为 0
alpha	标注的透明度	number	默认为 1，无透明，范围为 0～1
width	标注图标宽度	number/string	默认为图片实际宽度
height	标注图标高度	number/string	默认为图片实际高度
callout	自定义标记点上方的气泡窗口	Object	支持的属性见表 3.33，可识别换行符
label	为标记点旁边增加标签	Object	支持的属性见表 3.34，可识别换行符
anchor	经度和纬度在标注图标的锚点，默认底边中点		{x，y}，x 表示横向(0～1)，y 表示纵向(0～1)。{x：.5，y：1}表示底边中点

marker 上的气泡 callout，其属性如表 3.33 所示。

表 3.33　callout 属性

属　性　名	说　明	类　型	最低版本
content	文本	string	1.2.0
color	文本颜色	string	1.2.0
fontSize	文字大小	number	1.2.0
borderRadius	边框圆角	number	1.2.0
borderWidth	边框宽度	number	1.2.0
borderColor	边框颜色	string	1.2.0
bgColor	背景色	string	1.2.0
padding	文本边缘留白	number	1.2.0
display	'BYCLICK'：点击显示；'ALWAYS'：常显	string	1.2.0
textAlign	文本对齐方式，有效值：left、right、center、string	string	1.6.0

marker 上的气泡 label，其属性如表 3.34 所示。

表 3.34　label 属性

属　性　名	说　　明	类　　型	最低版本
content	文本	string	1.2.0
color	文本颜色	string	1.2.0
fontSize	文字大小	number	1.2.0
borderRadius	边框圆角	number	1.2.0
borderWidth	边框宽度	number	2.3.0
borderColor	边框颜色	string	2.3.0
bgColor	背景色	string	1.2.0
padding	文本边缘留白	number	1.2.0
display	'BYCLICK'：点击显示；'ALWAYS'：常显	string	1.2.0
textAlign	文本对齐方式，有效值：left、right、center	string	1.6.0

polyline 用于指定一系列坐标点，从数组第一项连线至最后一项，其属性如表 3.35 所示。

表 3.35　polyline 属性

属　性　名	说　　明	类　　型	备　　注
points	经度和纬度数组	array	[{latitude：0，longitude：0}]
color	线的颜色	string	十六进制
width	线的宽度	number	
dottedLine	带箭头的线	boolean	默认值为 false
arrowLine	边框宽度	number	默认值为 false，开发者工具暂不支持该属性
arrowIconPath	更换箭头图标	string	在 arrowLine 为 true 时生效
borderColor	线的边框颜色	string	
borderWidth	线的厚度	number	

circle 属性用于在地图上显示圆，通过设定中心点的经度、纬度和半径来绘制一个圆形图案作为地图上的标记物，其属性如表 3.36 所示。

表 3.36　circle 属性

属　性　名	说　　明	类　　型	备　　注
latitude	纬度	number	浮点数，范围为－90°～90°
longitude	经度	number	浮点数，范围为－180°～180°
color	描边的颜色	string	十六进制
fillColor	填充颜色	string	十六进制
radius	半径	number	
strokeWidth	描边的宽度	number	

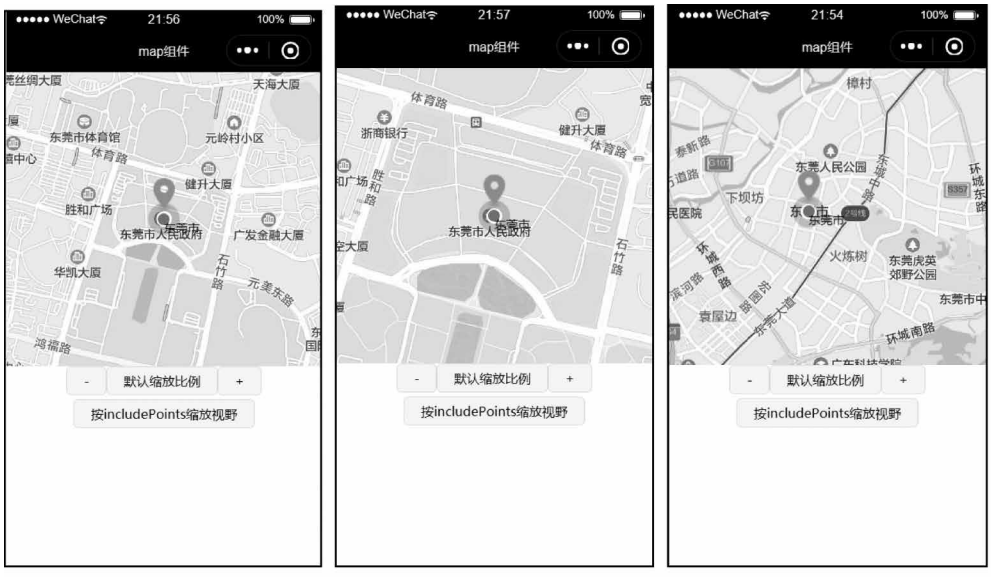
例 3-22　map 组件小案例，程序运行效果如图 3.33 所示。

视频讲解

pages/map/map.wxml 文件代码如下：

```
<map id="myMap" style="width: 100%; height: 300px" latitude="{{latitude}}" longitude="{{longitude}}" scale="{{scale}}" markers="{{markers}}" polyline="{{polyline}}" show-location></map>
```

```
<view class="content">
  <button size="mini" bindtap="reduce">-</button>
  <button size="mini" bindtap="default">默认缩放比例</button>
  <button size="mini" bindtap="add">+</button>
  </view>
  <view class="content">
   <button size="mini" bindtap="includePoints">按 includePoints 缩放视野</button></view>
```



(a) 16倍缩放比例效果　　(b) 17倍缩放比例效果　　(c) 包含两个点时的缩放比例效果

图 3.33　map 组件小案例

pages/map/map.js 文件代码如下：

```
Page({
  data: {
    latitude: 23.020670, longitude: 113.751790,
    scale:16,
    markers: [{
      latitude: 23.020670, longitude: 113.751790,
      iconPath: "/images/location.png",
      label: {
        content: "东莞市" }
    }],
    polyline: [{
      points: [{
        longitude: 113.3245211,
        latitude: 23.10229
      }, {
        longitude: 113.324520, latitude: 23.21229
      }],
      color: "#FF0000DD",
      width: 2,
      dottedLine: true
    }]
  },
```

```
  onReady: function(e) {
    this.mapCtx = wx.createMapContext("myMap")
  },
  default:function()
  {
    this.setData({ scale: 16, })
  },
  reduce: function() {
    this.setData({
      scale: this.data.scale - 1,
    })
  },
  add: function() {
    this.setData({
      scale: this.data.scale + 1,
    })
  },
  includePoints: function() {                          //缩放视野
    this.mapCtx.includePoints({
      padding: [10],
      points: [{
        latitude: 23.0403, longitude: 113.7446,
      }, {
        latitude: 22.9983, longitude: 113.7724,
      }]
    })
  }
})
```

【代码讲解】 本案例共定义 4 个函数：default、reduce、add 和 includePoints，其中，default 函数把地图设置为默认的 16 倍缩放比例，reduce 和 add 两个函数实现缩放比例的加一和减一，includePoints 函数指定(23.0403，113.7446)和(22.9983，113.7724)两个点，此时 map 地图缩放比例的依据就是需要在地图中包含这两个点，即这两个点需作为 map 地图 4 个顶点中的两个。

图 3.33(a)为默认的 16 倍缩放比例效果；图 3.33(b)为 17 倍缩放比例效果；图 3.33(c)为包含(23.0403，113.7446)和(22.9983，113.7724)两个点时的缩放比例效果。

3.8　实训项目——问卷调查

本实训项目是对微信小程序学习的问卷调查，知识点主要涉及表单组件的使用和表单数据在逻辑端 JS 中的获取。项目中涉及的表单组件有<form><radio-group><radio><checkbox-group><checkbox><label><slider><textarea>和<button>等。

视频讲解

项目在模拟器中的效果如图 3.34 所示；图 3.35 是 Console 控制台上显示的 JS 接收到的用户输入数据。

pages/survey/survey.wxml 文件代码如下：

```
<view class = "content">
```

```
<form bindsubmit="onSubmit" bindreset="onReset">
  <view class="title">1.你现在大几?</view>
  <radio-group bindchange="universityChange">
    <radio value="大一"/>大一
    <radio value="大二"/>大二
    <radio value="大三"/>大三
    <radio value="大四"/>大四
  </radio-group>
  <view class="title">2.你使用手机最大的用途是什么?</view>
  <checkbox-group bindchange="mobilChange">
    <label><checkbox value="社交"/>社交</label>
    <label>
      <checkbox value="网购"/>网购</label>
    <label>
      <checkbox value="学习"/>学习</label><label>
      <checkbox value="其他"/>其他</label>
  </checkbox-group>
  <view class="title">3.平时每天使用手机多少小时?</view>
  <slider min="0" max="24" show-value bindchange="timechange" />
   <view class="title">4.你之前使用过微信小程序吗?</view>
  <radio-group bindchange="programChange">
    <radio value="无"/>无
    <radio value="有"/>有
  </radio-group>
  <view class="title">5.谈谈你对微信小程序未来发展的看法</view>
  <textarea auto-height placeholder="请输入你的看法" name="textarea" />
  <button size="mini" form-type="submit">提交</button>
  <button size="mini" form-type="reset">重置</button>
</form>
</view>
```

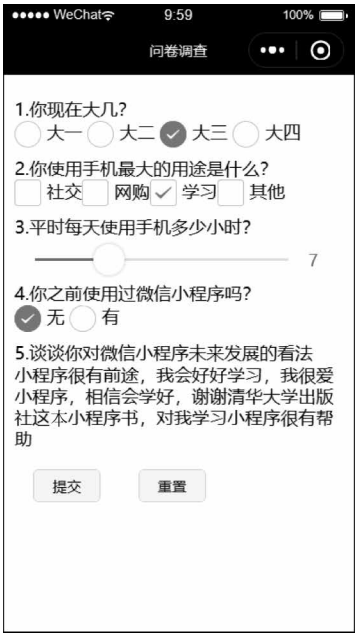


图 3.34　问卷调查示例图

图 3.35　Console 控制台显示数据

pages/survey/survey.js 文件代码如下：

```
Page({
  universityChange: function(e) {
    console.log("你选择的现在大几：", e.detail.value)
  },
  mobilChange: function(e) {
    console.log("你选择使用手机的最大用途是：", e.detail.value)
  },
  timechange: function(e) {
    console.log("你选择的每天使用手机的时间是：", e.detail.value + "小时")
  },
  programChange: function(e) {
    console.log("你选择的是否使用过微信小程序：", e.detail.value)
  },
  onSubmit(e) {
    console.log("你输入的对小程序发展前途的看法是" + e.detail.value.textarea)
  },
  onReset() {
    console.log("表单已被重置")
  }
})
```

pages/survey/survey.wxss 文件代码如下：

```
.content {
  padding: 30rpx;
}
button {
  margin: 40rpx;
}
.title {
  margin-top: 20rpx;
}
```

【代码讲解】 本项目是常用的问卷调查页面，“你现在大几？”和“你之前使用过微信小程序吗？”两项使用了< radio-group >和< radio >组件；“你使用手机最大的用途是什么？”使用了< checkbox-group >和< checkbox >组件；“平时每天使用手机多少个小时？”使用了< slider >组件；“谈谈你对微信小程序未来发展的看法”使用了< textarea >组件；“提交”“重置”按钮使用了< button >组件；而< form >和< label >组件属于控制类组件，没有在页面中显示效果。读者需认真学习 JS 页面获取表单数据的方法，为后续编程做准备。