第3章 Chapter 3

界面设计

前面讲到小程序的用户界面是由 WXML（页面布局文件）和 WXSS（布局样式文件）决定的。为了让开发者能够以最快的速度设计出美观而具有动态效果的小程序界面，本章结合实际案例的开发过程介绍微信小程序页面的常用样式和 flex 页面布局（弹性布局）。

本章学习目标

- 理解 CSS 标准规定的页面布局模型；
- 掌握样式在小程序界面设计中的使用方法；
- 掌握 flex 布局在小程序界面设计中的使用方法；
- 掌握 background-color、background-image、background-size、background-position 和 background-repeat 等样式属性在小程序界面设计中的使用方法；
- 掌握 view、text、input、button、swiper、image 和 scroll-view 等组件在小程序界面设计中的使用方法。

3.1 概述

早期的 Web 页面排版主要使用 HTML 中的 table（表格）元素实现，同时在 table 的单元格中通过 align、valign 属性来指定水平方向、垂直方向的对齐，但是这种方式很难直接传达页面的实际含义，使页面文档的可读性不高，且不容易维护。于是，1996 年 12 月，W3C（World Wide Web Consortium，万维网联盟）推出了用以解决结构与样式混杂问题的 CSS 规范的第一个版本，即 CSS1.0。1998 年，W3C 发布了 CSS 的第二个版本，即 CSS2.0。2001 年 5 月，W3C 完成了 CSS3 草案规范。CSS 是一种用来表现 HTML 或 XML 等文本样式的计算机语言，它不仅可以静态地修饰网页，配合各种脚本语言动态地对网页各元素进行格式化，还能够对网页中元素位置的排版进行像素级精确控制。CSS 标准规定的页面布局模型有以下 3 种：

3.1

（1）流动模型：流动模型是浏览器默认的一种网页布局模式，块级元素自上向下排列，行级元素自左向右排列。如表格布局就是采用流动模型实现页面元素的布局，它是 CSS 页面布局最简单的方式，页面元素排版布局时以默认的 html 元素在表格中的结构顺序显示。

（2）浮动模型：浮动模型提出的初衷是为了实现文字环绕效果，采用浮动布局的页面，浮动元素可以左右移动，直到它的边缘碰到父元素的边缘或另一个浮动元素的边缘。如浮动布局就是采用浮动模型实现页面元素的布局，它是目前 Web 页面设计采用最多的一种布局方式。

（3）层模型：层模型是根据页面元素的 position 属性值的不同来控制页面元素的显示位

置，position 的属性值有 static（静态）、absolute（绝对）、relative（相对）和 fixed（固定）四种定位布局。

这几种布局模型的搭配使用虽然可以比较轻松地完成页面设计的常见需求，但也存在缺少语义、不够灵活、浏览器兼容性差等缺陷。随着 Web 页面的语义化越来越流行，为了与搜索引擎建立良好的沟通、有助于爬虫和机器很好地解析网页内容，方便屏幕阅读器、盲人阅读器、移动终端设备等解析后渲染网页，W3C 从 CSS 的第三版本（CSS3.0）开始提出了弹性盒子模型（flex），该模型的 flex 布局方式可以更简便、完整、响应式地实现各种页面布局，目前也已经得到了几乎所有浏览器的支持。小程序的开发框架也使用了 flex 排版布局，它有助于开发者快速、灵活地构建小程序的 UI。

3.2 样式

微信小程序开发框架为开发人员提供了视图容器（View Container）、基础内容（Basic Content）、表单（Form）、导航（Navigation）、媒体（Media）、地图（Map）和画布（Canvas）、开放能力(Open Ability)等 8 大类组件，用于小程序的界面设计。但是，要达到用户想要的界面效果，还需要使用 WXSS 样式。WXSS 样式决定了组件应该如何显示，并在 CSS 样式的基础上做了一些功能扩展和修改。

3.2

3.2.1 长度单位

实现 CSS 样式时，开发者为了保证不同移动终端设备显示效果的适配性，需要考虑设备屏幕的不同宽度和像素比，采用一些技巧来换算像素单位。小程序的 WXSS 样式文件除了支持 CSS 的常用长度单位（表 3-1）外，还新增了一个新的长度单位——rpx（responsive pixel）。rpx 可以自动适配不同的屏幕宽度（规定屏幕宽度为 750rpx），开发者设计小程序的用户界面，如果使用 rpx 单位，换算工作就由小程序底层来完成，提高了开发效率。

表 3-1 CSS 常用长度单位

单位	说　明
pt	绝对长度单位，points（1pt 等于 1/72 英寸）
pc	绝对长度单位，picas（1pc 等于 12pt）
in	绝对长度单位，英寸
cm	绝对长度单位，厘米
mm	绝对长度单位，毫米
%	相对长度单位，百分比
px	相对长度单位，pixels 像素
em	相对长度单位，基于当前字体大写字母 M 的尺寸，当改变 font-size 的大小时，该长度单位将发生改变，默认 1em=16px
ex	相对长度单位，基于当前字体小写字母 x 的高度值

3.2.2 样式导入

小程序的 WXSS 支持样式的导入，这个功能在实际应用开发时非常有用，尤其是使用一些其他库的时候可以直接导入第三方的 WXSS 文件。用法步骤如下：

（1）新建要导入的 WXSS 样式文件，此处以新建的 common.wxss 样式文件为例，代码如下：

```
/** common.wxss **/
.top-p{
  padding:15px;
}
```

（2）使用@import 语句导入 WXSS 样式文件，此处以在 app.wxss 样式文件导入 common.wxss 样式文件为例，代码如下：

```
/** app.wxss **/
@import "common.wxss"; /** @import 语句后使用需要导入的样式文件的相对路径**/
.bottom-p {
  padding:10px;
}
```

3.2.3　内联样式与类样式

与 CSS 样式一样，WXSS 样式支持使用 style、class 属性控制组件的样式，但是实际使用时有区别：在小程序的样式使用时，对于静态样式，应统一定义到 WXSS 样式文件中，然后在 WXML 页面文件中由 class 属性设定；对于动态样式，需要直接在 WXML 页面文件中由 style 属性设定，这样有助于运行时动态解析，以便根据后台数据更新页面内容。如果将过多的样式直接使用 style 属性设定，会导致小程序在页面渲染时还要解析对应的样式布局，影响小程序渲染页面的速度。

style 接受动态样式，运行时会进行解析，尽量避免将静态的样式写入 style 中，以免影响渲染速度。代码格式如下所示：

```
<!-- style 属性控制样式 -->
<view style = "color:{{color}};"></view> <!-- color 需要在 js 文件中操作 -->
```

class 用于指定样式规则，通常用于统一样式类型的多个元素或重复使用的元素，减少重复代码的数量。其属性值是样式规则中类选择器名（样式类名）的集合。代码格式如下所示：

```
<!-- class 类控制样式 -->
<view class = "view_style" ></view>  <! -- view_style 需要在 wxss 文件中定义 -->
```

3.3　flex 布局

flex 是 flexible box（弹性盒子布局）的缩写，为页面使用盒状模型设计提供了最大的灵活性。开发设计微信小程序时，它既能符合小程序的文档开发要求，又能减少 CSS 的相关样式声明。

3.3

采用 flex 布局的元素称为 flex 容器（flex container），flex 容器中可以包含一个或多个 flex 容器项（flex item），一个容器项只能属于一个直接的容器，容器里面的多个容器项有排列方向。如图 3.1 所示，其中有一个 flex 容器（flex container）和三个容器项（flex item），这三个容器项从左到右排列，和容器项排列方向一致

的这条线称作主轴（main axis），图中的主轴为水平方向，与主轴垂直的一条线称作交叉轴（cross axis），图中的交叉轴为垂直方向。主轴的开始位置（与边框的交叉点，图 3.1 中左边框处）称为 main start，主轴的结束位置（与边框的交叉点，图 3.1 中右边框处）称为 main end；交叉轴的开始位置（与边框的交叉点，图 3.1 中上边框处）称为 cross start，交叉轴的结束位置（与边框的交叉点，图 3.1 中下边框处）称为 cross end。

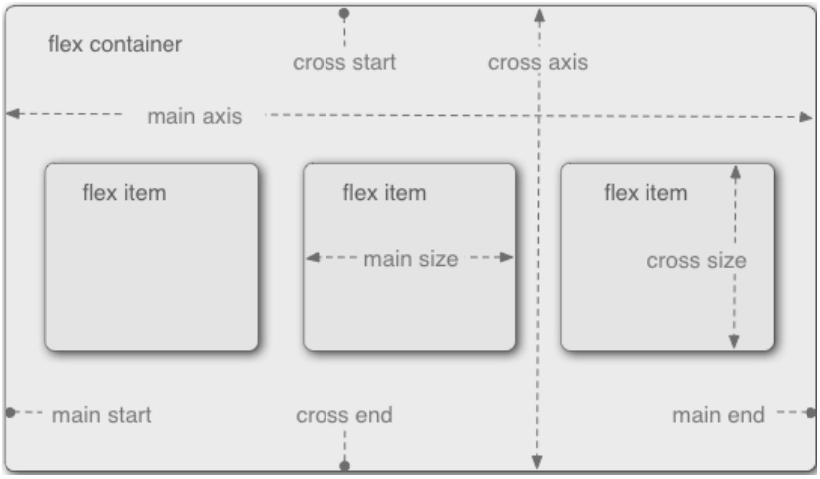

图 3.1　flex 布局结构

在小程序页面设计中，设有 display:flex 或 display:block 的元素就是一个 flex container，flex container 中的任何一个 flex item 也可以使用 block 或 flex 进行布局。

（1）display: block， 指定为块内容器模式，总是从新行开始显示容器项（flex item），小程序的视图容器（view、scroll-view、swiper、movable-view 和 cover-view 等）的样式属性 display 默认设置为 block。

（2）display: flex，指定为行内容器模式，默认在一行内显示容器项（flex item），也可以使用 flex-wrap 、flex-direction 等属性设置特定的显示效果。

例如，页面样式文件和页面结构文件分别如下列代码，其显示效果如图 3.2 所示。

（1）页面样式文件。

```
/** 页面样式文件——index.wxss **/
page {
  height: 100%;
  width: 100%;
  background-color: bisque;
}
.container {
  width: 100%;
  height: 100%;
  display: block;
}
.item {
  height: 80rpx;
  width: 80rpx;
  background-color: yellow;
  margin: 10rpx;
}
```

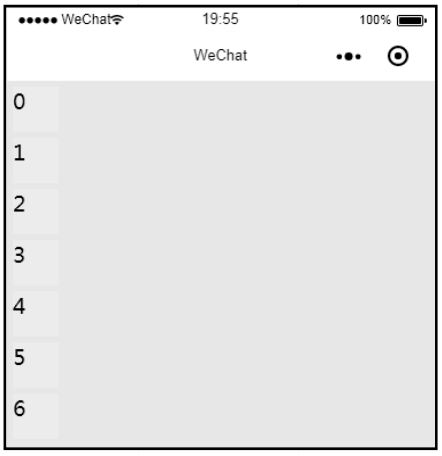

图 3.2　block 显示效果

用 CSS 设置样式时，width:100%会自动填满整个屏幕的宽度，而 height:100%只会自动适应子元素的高度。在正常的 HTML 代码中，要让子元素自动填充整个屏幕的高度，就必须给它的父元素也设置 height:100%，所以通常给网页的 HTML 或 body 属性增加 height:100%就可以了。由于在小程序页面结构文件（wxml 文件）中并没有 HTML 和 body 元素，所以不可能这样设置，但是小程序有一个 Page( )注册页面的方法，因此可以使用上述第 2~6 行的代码来设置 flex container 父容器（本示例的页面容器 page 即为父容器）的 height 属性，读者在开发小程序时务必要注意这一细节。

（2）页面结构文件。

```
1   <!-- 页面结构文件——index.wxml  -->
2   <view class="container">
3     <view class='item'>0</view>
4     <view class='item'>1</view>
5     <view class='item'>2</view>
6     <view class='item'>3</view>
7     <view class='item'>4</view>
8     <view class='item'>5</view>
9     <view class='item'>6</view>
10  </view>
```

如果将页面样式文件中第 10 行的 block 改为 flex，显示效果如图 3.3 所示。

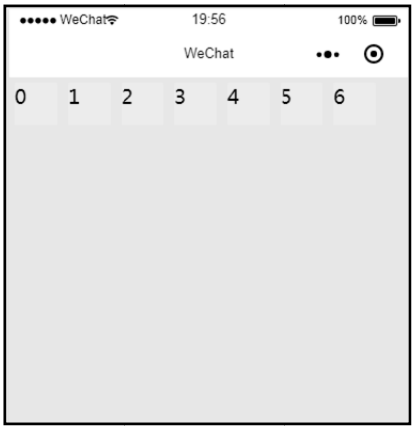

图 3.3　flex 显示效果

3.3.1 容器的属性

1 flex-direction

flex-direction 属性用来规定容器的主轴方向，主轴可以是水平方向，也可以是垂直方向，交叉轴垂直于主轴方向。即 flex-direction 属性用以指定容器项在容器中如何摆布，它有 4 个属性值。

（1）row（默认值）：主轴为水平方向，容器项的起点在水平方向的左端。例如，页面样式文件和页面结构文件分别如下列代码所示，其显示效果如图 3.3 所示。

① 页面样式文件。

```
/** 页面样式文件——flexdirection.wxss **/
page {
  height: 100%;
  width: 100%;
  background-color: bisque;
}
.container {
  width: 100%;
  height: 100%;
  display: flex;
  flex-direction: row;
}
.item {
  height: 80rpx;
  width: 80rpx;
  background-color: yellow;
  margin: 10rpx;
}
```

② 页面结构文件。

```
<!-- 页面结构文件——flexdirection.wxml  -->
<view class="container">
  <view class='item'>0</view>
  <view class='item'>1</view>
  <view class='item'>2</view>
  <view class='item'>3</view>
  <view class='item'>4</view>
  <view class='item'>5</view>
  <view class='item'>6</view>
</view>
```

（2）row-reverse：主轴为水平方向，容器项的起点在水平方向的右端。例如，将上述页面样式文件第 11 行代码的 flex-direction 属性值设置为 row-reverse，页面结构文件代码相同，其显示效果如图 3.4 所示。

（3）column：主轴为垂直方向，容器项的起点在垂直方向的上沿。例如，将上述页面样式文件第 11 行代码的 flex-direction 属性值设置为 column，页面结构文件代码相同，其显示效果如图 3.2 所示。

（4）column-reverse：主轴为垂直方向，容器项的起点在垂直方向的下沿。例如，将上述页面样式文件第 11 行代码的 flex-direction 属性值设置为 column-reverse，页面结构文件代码相同，其显示效果如图 3.5 所示。

图 3.4　row-reverse 显示效果

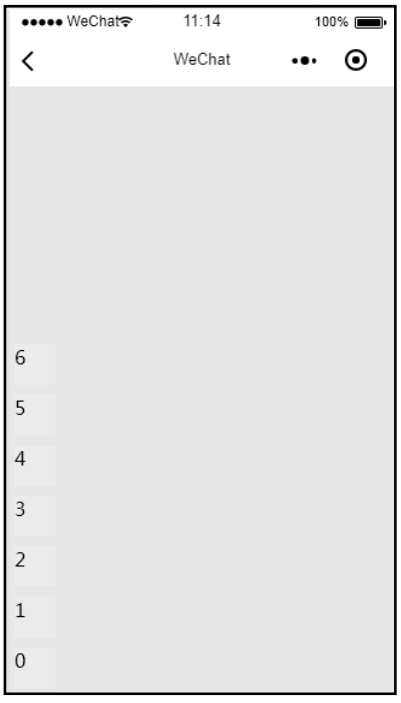

图 3.5　column-reverse 显示效果

2 flex-wrap

默认情况下，容器项都只能在一条主轴线上排列，超过界面宽度的容器项一般不显示。flex-wrap 属性用于指定容器项在一条主轴线上排列不下时如何换行（本部分介绍内容以主轴线在水平方向为例），它有如下 3 个属性值：

（1）nowrap（默认值）：容器项不换行，如果容器项超过容器的宽度和高度，会自动在主轴方向压缩；如果压缩后仍然超过容器的宽度和高度，超过的部分不会显示。例如，页面样式文件和页面结构文件代码如下，其显示效果如图 3.6 所示。

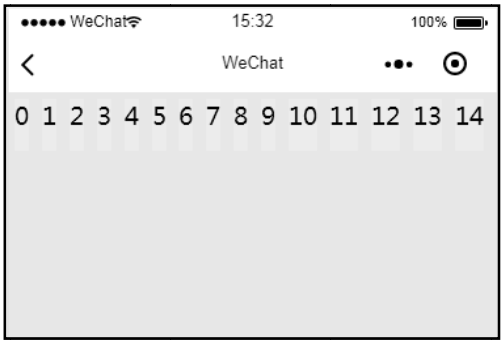

图 3.6　nowrap 显示效果

① 页面样式文件。

```
/** 页面样式文件——flexwrap.wxss **/
page {
  height: 100%;
  width: 100%;
  background-color: bisque;
```

```
}
.container {
  width: 100%;
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
}
.item {
  height: 80rpx;
  width: 80rpx;
  background-color: yellow;
  margin: 10rpx;
}
```

② 页面结构文件。

```
<!-- 页面结构文件——flexwrap.wxml  -->
<view class="container">
  <view class='item'>0</view>
  <view class='item'>1</view>
  <view class='item'>2</view>
  <view class='item'>3</view>
  <!--  与上面代码类似，此处略  -->
  <view class='item'>16</view>
  <view class='item'>17</view>
</view>
```

从图 3.6 可以看出，主轴是水平方向的，容器项没有换行，但每个容器项目的宽度已经在水平方向压缩，压缩后还有 14、15、16、17 等内容没有显示。

（2）wrap：容器项换行，从左到右排列，第二行在第一行的下面，第三行在第二行的下面，以此类推。例如，将上述页面样式文件第 11 行代码的 flex-wrap 属性值设置为 wrap，页面结构文件代码相同，其显示效果如图 3.7 所示。

（3）wrap-reverse：容器项换行，从左到右排列，第二行在第一行的上面，第三行在第二行的上面，以此类推。例如，将上述页面样式文件第 11 行代码的 flex-wrap 属性值设置为 wrap-reverse，页面结构文件代码相同，其显示效果如图 3.8 所示。

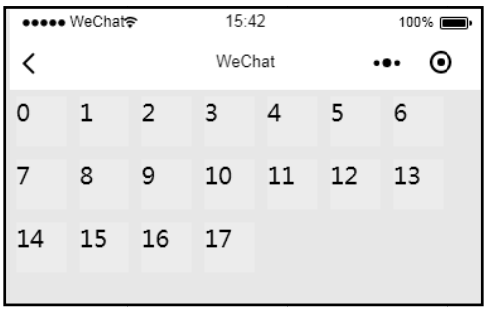

图 3.7 wrap 显示效果

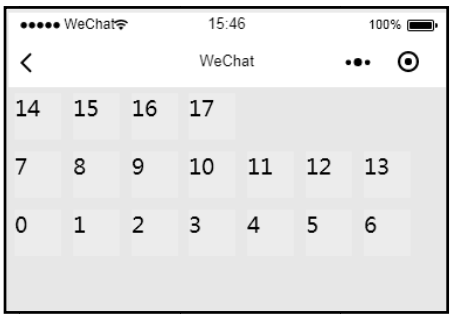

图 3.8 flex-flow 显示效果

3 flex-flow

flex-flow 属性是 flex-direction 和 flex-wrap 属性的简写形式，它的默认值是 row nowrap，即主轴为水平方向、不换行。flex-flow 属性值的设置可以使用下列形式：

```
flex-flow:flex-direction  flex-wrap ;
```

例如，要实现图 3.8 所示的显示效果，可以将前面 flexwrap.wxss 样式文件的第 10 行、第 11 行代码替换为下列代码：

```
1  flex-flow: row  wrap-reverse;
```

4 justify-content

justify-content 属性用于指定容器项在主轴方向的对齐方式，它有如下 5 个属性值：

（1）flex-start（默认值）：容器项与主轴的起始端对齐。如果主轴为从左向右的水平方向，即容器项左对齐；如果主轴为从上向下的垂直方向，即容器项顶端对齐。

（2）flex-end：容器项与主轴的末尾端对齐。如果主轴为从左向右的水平方向，即容器项右对齐；如果主轴为从上向下的垂直方向，即容器项底端对齐。例如，页面样式文件和页面结构文件代码如下，其显示效果如图 3.9 所示。读者需要注意图 3.5 与图 3.9 显示效果的区别，图 3.5 显示效果对应的主轴是从下向上的垂直方向，justify-content 的值默认为 flex-start，所以容器项与主轴的起始端对齐排列（从下向上的垂直方向排列）；而图 3.9 显示效果对应的主轴是从上向下的垂直方向，justify-content 的值设置为 flex-end，所以容器项与主轴的末尾端对齐排列。

图 3.9　flex-end 显示效果

① 页面样式文件

```
1   /** 页面样式文件——justifycontent.wxss **/
2   page {
3     height: 100%;
4     width: 100%;
5     background-color: bisque;
6   }
7   .container {
8     width: 100%;
9     height: 100%;
10    display: flex;
11    flex-direction：column;
12    justify-content：flex-end;
13  }
```

```
14  .item {
15    height: 80rpx;
16    width: 80rpx;
17    background-color: yellow;
18    margin: 10rpx;
19  }
```

② 页面结构文件

```
1   <!-- 页面结构文件——justifycontent.wxml  -->
2   <view class="container">
3     <view class='item'>0</view>
4     <view class='item'>1</view>
5     <view class='item'>2</view>
6     <view class='item'>3</view>
7     <!--  与上面代码类似，此处略   -->
8     <view class='item'>6</view>
9   </view>
```

（3）center：容器项在主轴方向排列时居中对齐。例如，将页面样式文件 justifycontent.wxss 的第 11 行和第 12 行修改为如下代码后，其显示效果如图 3.10 所示。

```
1    flex-direction：row;
2    justify-content：center;
```

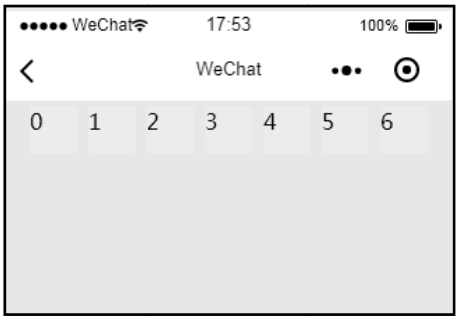

图 3.10　center 显示效果

（4）space-between：容器项沿主轴方向均匀分布，位于首尾两端的容器项与容器两端对齐。例如，将页面样式文件 justifycontent.wxss 的第 11 行和第 12 行修改为如下代码后，其显示效果如图 3.11 所示。

```
1    flex-direction：row;
2    justify-content：space-between;
```

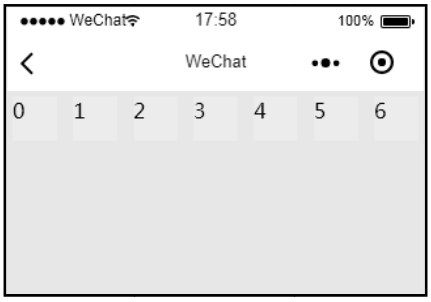

图 3.11　space-between 显示效果

（5）space-around：容器项沿主轴方向均匀分布，位于首尾两端的容器项到容器边缘的距离是容器项与容器项之间间距的一半。

5 align-items

align-items 属性用于指定容器项在交叉轴方向的对齐方式，它有如下 5 个属性值：

（1）stretch（默认值）：容器项沿交叉轴方向的尺寸拉伸至与容器一致。当主轴方向是水平方向时，如果容器项没有设置高度属性（height）或者高度属性值为 auto，容器项将会填满整个容器的高度。例如，页面样式文件和页面结构文件代码如下，其显示效果如图 3.12 所示。当主轴方向是垂直方向时，如果容器项没有设置宽度属性（width）或者宽度属性值为 auto，容器项将会填满整个容器的宽度。

① 页面样式文件。

```
/** 页面样式文件——alignitems.wxss **/
page {
  height: 100%;
  width: 100%;
  background-color: bisque;
}
.container {
  width: 100%;
  height: 100%;
  display: flex;
  flex-direction：row;
}
.item {
  width: 80rpx;
  background-color: yellow;
  margin: 10rpx;
}
```

② 页面结构文件。

```
<!-- 页面结构文件——alignitems.wxml  -->
<view class="container">
  <view class='item'>0</view>
  <view class='item'>1</view>
  <view class='item'>2</view>
  <view class='item'>3</view>
  <!--  与上面代码类似，此处略   -->
  <view class='item'>6</view>
</view>
```

（2）flex-start：容器项与交叉轴方向的起始端对齐。

（3）flex-end：容器项与交叉轴方向的末尾端对齐。

（4）center：容器项与交叉轴中点对齐。

（5）baseline：容器项与基线对齐。

6 align-content

align-content 属性用于指定多根轴线的对齐方式。如果容器项目只有一根轴线，该属性不起作用。它共有 flex-start、flex-end、center、space-between、space-around 和 stretch（默认值）等 6 个属性值，其功能含义与前面介绍的一样，这里不再赘述。

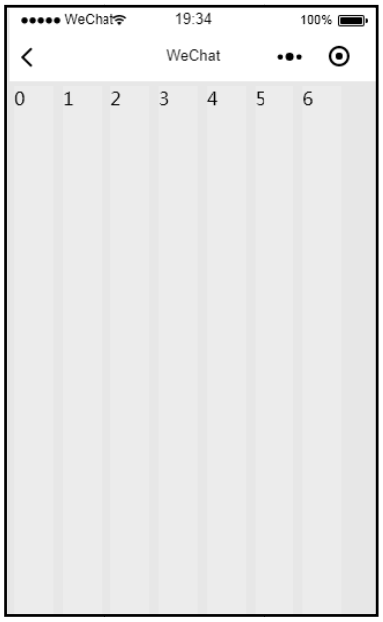

图 3.12　stretch 显示效果

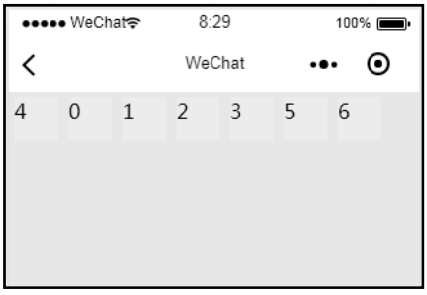

图 3.13　order 显示效果

3.3.2　容器项的属性

1 order

容器项排列的位置默认是按照页面结构文件代码中的先后顺序排列的，也就是哪个容器项写在前面，默认就排列在前面。而 order 属性可以改变容器项的排列顺序，数值越小，排列越靠前，默认值为 0。例如，页面样式文件和页面结构文件代码如下，其显示效果如图 3.13 所示。

① 页面样式文件。

```
/** 页面样式文件——order.wxss **/
page {
  height: 100%;
  width: 100%;
  background-color: bisque;
}
.container {
  width: 100%;
  height: 100%;
  height: 100%;
  display: flex;
  flex-direction: row;
}
.item {
  width: 80rpx;
  height: 80rpx;
  background-color: yellow;
  margin: 10rpx;
}
.item:nth-child(5) {
  order: -1;
}
```

② 页面结构文件。

```
<!-- 页面结构文件——order.wxml  -->
<view class="container">
  <view class='item'>0</view>
  <view class='item'>1</view>
  <view class='item'>2</view>
  <view class='item'>3</view>
  <!--  与上面代码类似，此处略   -->
  <view class='item'>6</view>
</view>
```

上述代码的第 20 行使用了 nth-child(n)选择器，表示匹配属于其父元素的第 n 个子元素，此处 item：nth-child(5)规定属于容器的第 5 个容器项的 order 属性值为−1。从图 3.12 的显示效果可以看出，第 5 个容器项就是用 view 组件显示的“4”元素，其 order 属性值被设置为−1，其余元素的 order 属性值仍然是默认值 0，所以 4 元素通过这样设置后，排在其他元素的最前面。

2 flex-grow

flex-grow 属性用于定义容器项的放大比例，默认值为 0，即如果存在剩余空间，该容器项也不会放大显示。如果所有容器项的 flex-grow 属性值设为 1，则它们将等分剩余空间；如果其中一个容器项的 flex-grow 属性值设为 n，其他容器项目都设为 1，则前者占据的剩余空间将比其他容器项多 n 倍。当有放大空间时，值越大，放大的比例越大。

3 flex-shrink

flex-shrink 属性用于定义容器项的缩小比例，默认值为 1，即如果空间不足，该容器项目将缩小，值越大，缩小的比例越小。如果所有容器项目的 flex-shrink 属性值设为 1，当空间不足时，都将等比例缩小；如果一个容器项的 flex-shrink 属性值设为 0，其他容器项都设为 1，当空间不足时，前者不缩小。负值对该属性无效。

4 flex-basis

flex-basis 属性用于定义分配多余空间之前容器项占据的主轴空间。小程序根据这个属性计算主轴是否有多余空间，默认值为 auto，即容器项的本来大小。例如，如果容器项目有多余的空间，flex-basis 属性设置为 200rpx，那么容器项会放大到 200rpx 的宽度。

5 flex

flex 属性是 flex-grow、flex-shrink 和 flex-basis 属性的简写，默认值为 0、1、auto，后两个属性可选。该属性有两个快捷值：auto(1 1 auto)和 none(0 0 auto)。建议使用时优先使用这个属性，而不是单独写 3 个分开的属性值，这样有助于提高小程序的效率。

6 align-self

align-self 属性用于设置单个容器项独特的对齐方式，默认值为 auto（表示继承父容器的 align-items 属性）；该属性可以取 6 个值，除了 auto 值外，其他都与 align-items 属性完全一致。

3.4　仿“猜画小歌”界面设计

“猜画小歌”是 Google 于 2018 年 7 月 18 日发布的首款微信小程序。该小程序可以让每个人都有机会体验人工智能技术驱动下的人机交互。本节将模仿“猜画小歌”小程序首个界面（见图 3.14）的实现过程介绍 view 组件、text 组件和背景、颜色、边框等样式在小程序界面设计中的使用方法。

3.4.1

图 3.14 “猜画小歌”界面

3.4.1 预备知识

1 view

view（视图组件）是一个容器组件，是微信小程序界面设计中最常用、最基础的组件。它里面不仅可以放置其他组件，还可以直接显示文本信息。本章 3.3 节已经在 flex 布局中阐述了用 view 组件显示文本信息和小程序界面的布局方法。该组件与 HTML 中的 div 标签一样，可以实现页面结构的划分、页面布局的调整等，另外还有表 3-2 中所示的单击态属性和功能。

表 3-2 view 组件的属性及功能说明

属性名	类型	说　明
hover-class	String	指定当单击 view 组件时调用的样式类。默认值为 none，即没有单击效果
hover-stop-propagation	Boolean	指定是否阻止本节点的祖先节点出现单击态，即当单击本节点时，是否调用祖先节点 hover-class 指定的样式类。特别说明，只要在代码中定义了该属性，不管属性值是 true 还是 false，都是阻止的
hover-start-time	Number	指定按住后多久出现调用 hover-class 指定的样式类。默认值为 50，单位为 ms
hover-stay-time	Number	指定松开后调用 hover-class 指定的样式类状态保留时间。默认值为 400，单位为 ms

以下面一段代码为例，介绍表 3-2 中 4 个属性的用法。

页面结构代码如下：

```
1   <view hover-class="red"  class="view">单击 view 才会调用.red 样式类</view>
2   <view hover-class="blue">
3     <view>其他内容</view>
4     <view hover-stop-propagation="true" hover-class="yellow">单击此 view 阻止
调用.blue 样式类</view>
5   </view>
6   <view hover-start-time="1000" hover-class="red">单击 view 1 秒后调用.red 样式
类</view>
7   <view hover-stay-time="3000"  hover-class="red">单击 view 能让.red 样式持续 3
秒</view>
```

页面样式代码如下：

```
1   .red{
2     color: red;
3   }
4   .blue{
5     color: blue;
6   }
7   .yellow{
8     color: yellow;
9   }
```

页面结构代码第 1 行指定了 view 组件的 hover-class 属性，当单击 view 时，会调用.red 样式类，即此时 view 组件上显示的文本为红色；第 2~5 行使用了 view 嵌套来显示文本，第 4 行中使用了 hover-stop-propagation 属性，此时若单击第 4 行定义的 view，会调用.yellow 样式类，但不会调用其祖先节点（第 2 行定义的 view）定义的.blue 样式类，若没有使用 hover-stop-propagation 属性，则不仅调用.yellow 样式类，还调用其祖先节点定义的.blue 样式类。

2 text

text（文本组件）是小程序中最基础的组件之一，常用于在页面中展示文字，它除了包含诸如颜色、背景等基础组件的属性外，还包括表 3-3 所示的小程序特有属性。

表 3-3　text 组件属性及功能说明

属性名	类型	说　明
selectable	Boolean	文本是否可选，默认值为 false（文本不可选）
space	String	显示连续空格，默认不显示连续空格
decode	Boolean	是否解码，默认值为 false（不解码）

text 支持\n 换行；selectable 属性用于指定 text 显示的文本在页面上是否可选，即当 selectable 为 true 时，可以长按文本选中，也可以复制。这两个特性是 text 与 view 组件在显示文本内容时的区别。

space 属性用于设置文本中的空格在页面上显示时的间距，它有 ensp（空格显示时为中文字符空格一半大小）、emsp（空格显示时为中文字符空格大小）和 nbsp（空格显示时根据字体设置空格大小）三个属性值。例如，space 的属性值为 emsp，表示 text 中的文本若有多个空格，则在页面上以连续多个中文字符空格显示。

decode 属性用于设置是否可以在页面上渲染一些特殊的标签，例如小于号<（&lt）、大于号>（&gt）、与符号&（&amp）、撇号'（&apos）及空格号（ 、 、 ）。默

认状态下，decode 的属性值为 false，即这些符号在小程序中默认不能被直接渲染显示，如果要渲染显示，就需要将 decode 属性值设为 true。例如，以下代码表示 text 中的文本不可选，文本中的空格根据字体设置间距大小，可以在页面上显示<和>等特殊符号。

```
<view>
  <text selectable='false' space='nbsp' decode='true'>
   This is the first line.\n&lt;&gt;
  </text>
</view>
```

下面使用 view 和 text 组件及样式实现图 3.15 所示的界面，实现代码如下。

① 页面结构文件。

```
<view class="bcontainer">
  <view class="view_name">
    <text>人的一生至少要有两次冲动</text>
    <text>一次为奋不顾身的爱情</text>
    <text>一次为奋不顾身的爱情</text>
    <text>一次为说走就走的旅行</text>
    <text class="text_line">——</text>
    <text>Andy Andrews</text>
  </view>
</view>
```

在以上代码中，最外层使用 view 组件作为容器，控制图 3.15 界面的全部背景效果，并设置了 class 的属性，即使用了类选择器.bcontainer，.bcontainer 样式需要在页面样式文件中定义；然后在内层使用 view 组件放置具体的文本显示内容，由于文本显示内容需要指定样式，所以本案例的文本显示内容使用了 text 组件，也使用了类选择器.view_name，用于控制整个文本显示区域的背景。代码的第 3~8 行中分别定义了 text 组件上显示的内容，从图 3.15 可以看出只有第 7 行内容格式与其他行的显示效果有差别，所以使用了类选择器.text_line。

② 页面样式文件。

```
page {
  width: 100%;
  height: 100%;
}
.bcontainer {
  height: 100%;
  background: #d0dbec;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
}
.view_name {
  width: 100%;
  background-color: #a6d1ca;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
text {
  font-size: 25rpx;
  align-self: center;
}
.text_line {
```

```
25    padding: 25rpx;
26  }
```

以上代码的第 5~12 行定义了外层 view 的布局，其中第 8~9 行设置布局使用主轴为垂直方向（从上至下）的 flex 布局、第 10 行设置布局中的内容在交叉轴方向（水平方向）居中、第 11 行设置布局中内容沿主轴方向（垂直方向）均匀分布；第 13~19 行定义了内层 view 的布局，其中第 16~17 行设置布局使用主轴为垂直方向（从上至下）的 flex 布局、第 18 行设置布局中内容（即 text 组件）沿主轴方向（垂直方向）居中对齐；第 20~23 行定义了一个控制 text 组件样式的元素选择器，用于设置 text 组件上显示的文字大小和对齐方式；第 24~26 行定义了图 3.15 中第 5 行文本的内边间距。

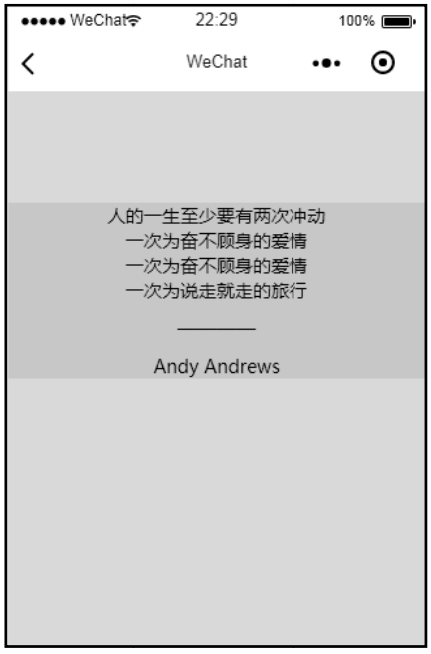

图 3.15　view 和 text 的显示效果

3 背景与颜色

CSS 允许应用颜色值作为背景，也允许使用背景图像创建复杂的效果。小程序的样式文件基本延续了 CSS 的相关属性，但是实际使用中还是有一些差异的。下面介绍小程序界面设计中常用的一些属性作用和使用方法。

（1）background-color。

background-color 属性用于为页面元素设置一种背景颜色，该颜色会填充元素的内容、内边距和内边框。background-color 的属性及功能如表 3-4 所示。

表 3-4　background-color 的属性及功能说明

值	说　明
color_name	规定颜色值为颜色名称的背景颜色（如 red）
hex_number	规定颜色值为十六进制值的背景颜色（如#ff0000）
rgb_number	规定颜色值为 rgb 代码的背景颜色（如 rgb（255,0,0））

续表

值	说　明
transparent	默认。背景颜色为透明
inherit	规定应该从父元素继承 background-color 属性的设置

（2）background-image

background- image 属性用于为页面元素设置一个背景图像，该图像占据元素的全部尺寸，包括内边距和内边框，默认背景图像位于元素的左上角，并在水平和垂直方向上重复显示。background-image 的属性及功能如表 3-5 所示。

表 3-5　background-image 的属性及功能说明

值	说　明
url（'URL'）	指向图像的路径
none	默认值。不显示背景图像
inherit	从父元素继承 background-image 属性的设置

需要说明的是：本地资源图片无法通过微信小程序的样式文件获取，所以在微信小程序设计时，如果需要使用 background-image 属性设置背景图片，可以在样式文件中使用网络图片。例如，要实现图 3.16 的显示效果，可以使用如下代码。

① 页面结构文件。

```
<view class="container">
  <view class="view_img"></view>
  <view class="view_name">
    <text >人的一生至少要有两次冲动</text>
    <!--  与上面代码类似，此处略  -->
    <text class="text_line">————</text>
    <text >Andy Andrews</text>
  </view>
</view>
```

从上述代码可以看出，实现图 3.16 的效果是在图 3.15 的基础上增加了第 2 行代码，即使用 1 个 view 组件和定义了 1 个 view_img 样式选择器显示图片。

② 页面样式文件。

```
<!--  与上面代码类似，此处略  -->
.bcontainer {
  height: 100%;
  background: #d0dbec;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
}
<!--  与上面代码类似，此处略 以下为新增样式定义  -->
.view_img{
  width: 230rpx;
  height: 230rpx;
  background-image: url('https://zsc.nnutc.edu.cn/__local/B/BC/58/
BE88E69F69A86F324E5122302B9_8E7373D8_48D3D.jpg');
  background-size: cover;
}
```

上述代码的第 11~16 行定义了.view_img 类选择器设置页面样式，width、height 属性分别指定 view 组件的宽度、高度，background-image 属性指定了 view 组件的背景图片，background-size 属性值为 cover，表示让图片等比例缩放，铺满整个 view。如果要实现图 3.17 所示的图片效果，可以在上述代码.view_img 类选择器代码中添加“border-radius: 50%;”，border-radius 属性用于给 view 设置圆角边框。

图 3.16　background-image 显示效果

图 3.17　border-radius 的显示效果

但是，如果 view 组件中显示的是本地图片资源，且不在样式文件中设置，则只可以在定义组件时使用内联样式，即用下面代码替换上述页面结构的第 2 行代码，并删除上述页面样式代码的第 14 行。

```
1    <view class="view_img" style='background-image:url(../../images/lvxin.
jpg);'></view>
```

（3）background-size。

background-size 属性用于设定背景图像的尺寸，它的属性及功能如表 3-6 所示。

表 3-6　background-size 的属性及功能说明

值	说　明
auto	背景图的实际大小，默认值
length	设置背景图像的高度和宽度。第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto。如 background-size:100px 100px;
percentage	以父元素的百分比来设置背景图像的宽度和高度，第一个值设置宽度，第二个值设置高度。如果只设置一个值，则第二个值会被设置为 auto。如 background-size:100% 100%;
cover	等比例缩放背景图像以完全覆盖背景区域，有可能超出背景区域

续表

值	说　明
contain	将背景图片等比例缩放到宽度或高度与背景区域的宽度或高度相等，背景图片始终在背景区域内

（4）background-position。

background-position 属性用于设定背景图像的起始位置，它的属性及功能如表 3-7 所示。

表 3-7　background-position 的属性及功能说明

值	说　明
xpos、ypos	水平方向可选 left、center、right，垂直方向可选 top、center、bottom；xpos 为 left 表示图像的左边与对象的左边对齐，为 right 表示图像的右边和对象的右边对齐；ypos 为 top 表示图像的顶部和对象的顶部对齐，为 bottom 表示图像的底部和对象的底部对齐；xpos、ypos 为 center 表示图像在水平和垂直方向的中心和对象在水平和垂直方向的中心对齐；如果仅指定一个值，则另一个值是 center
x%、y%	第一个值是水平位置，第二个值是垂直位置。左上角是 0、0。右下角是 100%、100%。如果仅指定了一个值，其他值将是 50%。 默认值为 0、0
x、y	第一个值是水平位置，第二个值是垂直位置。左上角是 0、0。单位可以是像素（0px，0px）或任何其他单位。如果仅指定了一个值，其他值将是 50%。默认值为 0、0
inherit	从父元素继承 background-position 属性的设置

（5）background-repeat。

background-repeat 属性用于设定背景图像是否重复或如何重复，它的属性值及功能如表 3-8 所示。

表 3-8　background-repeat 的属性及功能说明

值	说　明
repeat	背景图像向垂直和水平方向重复。默认值
repeat-x	只有水平位置会重复背景图像
repeat-y	只有垂直位置会重复背景图像
no-repeat	背景图像不会重复
inherit	从父元素继承 background-repeat 属性的设置

3.4.2　仿“猜画小歌”界面的实现

1 素材准备

根据图 3.14 的显示效果，需要准备界面背景图片 bground.jpg、头像图片 touimg.jpg、操作按钮图片 btn1.jpg、btn2.jpg、btn3.jpg 和 btn4.jpg，并将这些图片保存到项目的 images/guess 文件夹下。

3.4.2

2 界面实现

本案例实现时，将图 3.14 整个界面的设计过程分为背景、头像、用户信息和操作按钮四个部分，如图 3.18 所示。

（1）背景。

① 页面布局代码。

```
1   <view class="container" style="background-image: url('../../images/guess/
```