第 3 章

界 面 设 计

HarmonyOS 应用程序的 Java 工程通常用 xml 格式的页面布局文件设计用户界面，HarmonyOS 应用程序的 JS 工程通常用 hml 格式的页面布局文件和 css 格式的页面样式文件设计用户界面。为了让开发者能够以最快的速度设计出美观、具有动态效果的 JS 工程用户界面，本章结合实际案例的开发过程介绍 JS 工程的常用样式和 flex 弹性布局。

3.1 样式

css 是一种用来表现 html 或 xml 等文档样式的语言。css 不仅可以静态地修饰用 html 或 xml 标识的页面，还可以配合各种脚本语言动态地对页面上的各个元素进行格式化。css 能对页面中元素位置的排版进行像素级精确控制，支持几乎所有的字体、字号样式，拥有对页面对象和模型样式编辑的能力。为了方便应用程序的页面开发，在 HarmonyOS 应用程序开发框架中引入了传统 css 的大部分特性，同时为了更适合开发 HarmonyOS 应用程序，也对传统的 css 进行了扩充和修改，让 css 修饰的 hml 页面结构更丰富、更灵活。

扫一扫

3.1.1 尺寸单位

1. 逻辑像素 px

应用程序配置文件(config.json)的 window 选项用于定义与显示窗口相关的配置，其代码格式如下。

```
{
      ......
"window": {
          "designWidth": 720,
          "autoDesignWidth": true
        }
        ...
}
```

designWidth 属性用于指定屏幕逻辑宽度。默认状态下，手机和智慧屏设备的逻辑宽

度为 720px，智能穿戴设备的逻辑宽度为 454px。所有与尺寸大小相关的样式（如 width、font-size）都以 designWidth 属性值和实际屏幕宽度的比例进行缩放。例如，如果在配置文件中指定 designWidth 属性值为 720，而样式文件中设置的 width 值为 100px，则在实际宽度为 1440 物理像素的设备屏幕上，width 的实际渲染像素值为 200 物理像素，也就是从 720px 到 1440 物理像素，所有尺寸大小放大 2 倍。

autoDesignWidth 属性用于指定渲染组件和布局时是否按屏幕密度进行缩放，如果该属性值为 true，则设置的 designWidth 属性值会被忽略，屏幕逻辑宽度由设备宽度和屏幕密度自动计算得出，在不同设备上可能不同。例如，样式文件中设置的 width 值为 100px，在屏幕密度为 3 的设备上，实际渲染像素值为 300 物理像素。

2. 百分比

组件的尺寸单位用百分比表示时，表示该组件占父组件尺寸的百分比。例如，如果组件的 width 属性值设置为 50%，则代表该组件渲染时在页面上的宽度为其父组件宽度的一半。

3.1.2　通用样式


扫一扫

HarmonyOS 应用程序开发框架所支持的通用样式与传统 css 样式类似，为方便读者阅读和理解本书的范例代码，表 3.1 中列出了部分常用的通用样式和功能说明。

表 3.1　常用的通用样式和功能说明

样式属性	功能描述	示例代码
color	设置前景色	例如，color：yellow 表示设置前景色为黄色
background-color	设置背景色	例如，background-color：yellow 表示设置背景色为黄色
font-size	设置字体大小	例如，font-size：25px 表示设置字体大小为 25px
border	设置边框线	例如，border：5px solid red 表示设置边框线宽度为 5px、红色
margin	设置外边距	例如，margin：15px 5px 25px 10px 表示设置上、右、下和左侧的外边距
padding	设置内边距	例如，padding：15px 5px 25px 10px 表示设置上、右、下和左侧的内边距
width	设置宽度	例如，width：50%表示设置宽度占容器宽度的 50%
height	设置高度	例如，height：200px 表示设置高度为 200px

在 css 中，可以通过使用预定义的颜色名称、RGB、RGBA 或 HEX 的值指定颜色。例如，red、rgb(255，0，0)、＃FF0000 等颜色值表示指定颜色为红色，rgba(255，0，0，0.5)、＃55FF0000 等颜色值表示指定颜色为透明红色。

3.1.3　样式使用


扫一扫

在页面布局文件中，通常用 style、class 属性控制组件的样式。style 属性声明的样式也

称为内联样式。应用程序的每个页面目录下都有一个与页面布局文件同名的样式文件，用来描述该页面布局文件中组件的样式，决定组件应该如何显示。

【范例 3-1】 用 style 属性设置 text 组件的前景色为红色，背景色为黄色。

hml 的代码如下：

```
<div>
    <text style="color: red;background-color: yellow;">HarmonyOS 应用开发
</text>
</div>
```

【范例 3-2】 用 class 属性设置 text 组件的前景色为红色，背景色为黄色。

css 的代码如下：

```
.textcolor{
    color: red;
    background-color: yellow;
}
```

hml 的代码如下。

```
<div>
    <text class="textcolor">HarmonyOS 应用开发</text>
<div>
```

3.1.4 动画样式

HarmonyOS 应用程序开发框架从 API version 4 开始支持动画效果 hml 元素，动画效果可以让 hml 元素逐渐从一种样式变为另一种样式。在使用 css 样式设置动画时，必须首先用@keyframes 为动画指定一些关键帧。表 3.2 列出了部分常用的动画样式属性和功能说明。

表 3.2 常用的动画样式属性和功能说明

样式属性	功能描述	示例代码
animation-name	设置动画关键帧名称	例如，animation-name：example 表示设置动画关键帧名称为 example，example 必须在样式文件中已定义
animation-delay	设置动画播放的延迟时间	例如，animation-delay：4s 表示设置动画在 4s 后播放，单位为 ms(毫秒，默认)、s(秒)
animation-duration	设置动画播放的持续时间	例如，animation-duration：4s 表示设置动画播放持续 4s，单位为 ms(毫秒，默认)、s(秒)
animation-iteration-count	设置动画播放的次数	例如，animation-iteration-count：infinite 表示设置动画播放无限次，默认播放 1 次

续表

样式属性	功能描述	示例代码
transform	设置平移/旋转/缩放等动画效果	例如，transform：translateX(250px)表示设置沿 X 轴方向移至 250px 的平移动画；例如，transform：rotate(90deg)表示设置顺时针旋转 90°的旋转动画；例如，transform：scaleY(4)表示沿 Y 轴方向放大至 4 倍的缩放动画；例如，transform：skewX(80deg)表示设置逆时针倾斜 80°的倾斜动画

【范例 3-3】 设置显示 Hello World! 的 text 组件背景色从红色变为黄色，并且具有从 0°～90°～180°顺时针旋转和逐渐放大的动画效果。

hml 的代码如下。

```
<div class="container">
    <text class="title">
        Hello World!
    </text>
</div>
```

css 的代码如下。

```
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 454px;
    height: 454px;
}
.title {
    background-color: red;
    width: 200px;
    height: 200px;
    animation-name: example;                    /* 指定动画关键帧名称 */
    animation-duration: 4s;                     /* 动画播放时间为 4 秒 */
    animation-iteration-count: infinite;        /* 动画播放无限次 */
    text-align: center;
}
@keyframes example {                            /* 定义关键帧，名称为 example */
    from {                                      /* 开始关键帧样式 */
        background-color: red;
        transform: rotate(0deg) scale(1.0);
    }
    50% {                           /* 通过百分比指定动画运行的中间状态关键帧样式 */
        background-color: white;
        transform: rotate(90deg) scale(1.2);
    }
```

```
    to {                                          /*终止关键帧样式*/
        background-color: yellow;
        transform: rotate(180deg) scale(1.5);
    }
}
```

上述第 18～21 行代码定义动画起始帧的 text 组件背景色为红色、旋转角度为 0°、放大倍数为 1 倍；第 22～25 行代码定义动画运行至 50%时的关键帧样式，也可以不定义；第 26～29 行代码定义动画终止帧的 text 组件背景色为黄色、旋转角度为 180°、放大倍数为 1.5 倍。

3.1.5 渐变样式

HarmonyOS 应用程序开发框架从 API version 4 开始支持线性渐变(linear-gradient)和重复线性渐变(repeating-linear-gradient)两种渐变效果，实现两个或多个指定颜色间的平稳过渡。使用渐变样式，需要定义过渡方向和过渡颜色，也就是要指定 direction 参数、angle 参数和 color 参数的值，表 3.3 列出了这 3 个参数的使用说明。

表 3.3 direction、angle 和 color 参数的使用说明

参数名称	默认值	使用说明
direction	to bottom(由上至下渐变)	指定过渡方向，参数值格式：to[left|right]||[top|bottom]。如：to left(从右向左渐变)，to bottom right(从左上角到右下角渐变)
angle	180deg(180°)	指定过渡方向，以组件几何中心为坐标原点，水平方向为 X 轴，angle 指定了渐变线与 Y 轴顺时针方向的夹角
color	无	指定使用渐变样式区域内颜色的渐变效果，至少指定两种颜色，参数值格式：#FF0000、#FFFF0000、rgb(255,0,0)或 rgba(255,0,0,1)

【范例 3-4】 设置显示 Hello World! 的 text 组件在距离左边 27px 和距离左边 227px(454×0.5)之间 200px 宽度形成渐变效果。

将范例 3-3 的 css 代码替换为如下代码。

```
1  .title {
2      font-size: 30px;
3      text-align: center;
4      width: 200px;
5      height: 200px;
6      background: linear-gradient(to right, rgb(255, 0, 0) 27px, rgb(0, 255, 0)
50%);
7  }
```

上述第 6 行代码 linear-gradient()方法用于设置线性渐变，to right 表示渐变方向从左向右，rgb(255, 0, 0) 27px 表示距左边 27px 开始为红色效果，rgb(0, 255, 0) 50%表示距左边 227px(454×50%)处开始渐变为绿色效果。如果要实现从左向右重复渐变，重复渐变区域 30px(60px-30px)，则需要将上述第 6 行代码修改为如下代码。

```
1   background: repeating-linear-gradient(to right, rgb(255, 255, 0) 30px,rgb
(0, 0, 255) 60px);
```

3.2　flex 布局


扫一扫

flex 的英文全称为 Flexible Box，中文含义为弹性盒子布局，它是一种布局模型。采用 flex 布局的元素，称为 flex 容器（flex container，简称“容器”）。flex 容器中的所有子元素，称为 flex 项目（flex item，简称“项目”）。flex 布局的坐标系是以容器左上角的点为原点，自原点向右、向下有两个坐标轴，即一个主轴（main axis）和一个交叉轴（cross axis），交叉轴与主轴互相垂直。主轴的开始位置（与边界的交叉点）称为主轴起点（main start），主轴的结束位置称为主轴终点（main end）；交叉轴的开始位置（与边界的交叉点）称为交叉轴起点（cross start），交叉轴的结束位置称为交叉轴终点（cross end）。项目默认沿主轴方向排列，单个项目占据的主轴空间称为 main size，单个项目占据的交叉轴空间称为 cross size。flex 默认布局中自原点向右的坐标轴为主轴，自原点向下的坐标轴为交叉轴，如图 3.1 所示。

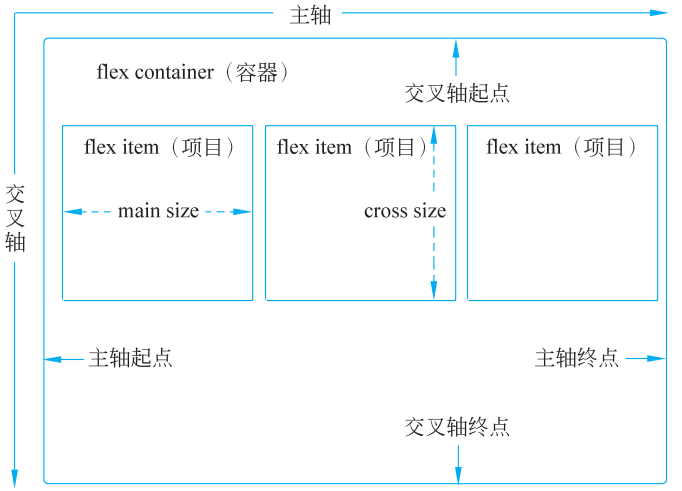

图 3.1　flex 默认布局模型

3.2.1　容器的属性

1. flex-direction 属性

flex-direction 属性用于设置容器的主轴方向，主轴方向决定了项目在容器中的摆布方式，它的属性值包含 row 和 column。

（1）row：主轴为自左向右的水平方向，默认值，即项目按水平方向（行方向）从左到右排列。

(2) column：主轴为自上向下的垂直方向，即项目按垂直方向(列方向)从上到下排列。

【范例 3-5】 用 flex-direction 属性实现如图 3.2 所示效果。

css 的代码如下。

```
.container {
    display: flex;
    flex-direction: row;
    width: 100%;
    height: 100%;
}
.item {
    background-color: blueviolet;
    border: 1px solid yellow;
    font-size: 20px;
    text-align: center;
    width: 80px;
    height: 100px;
}
```

hml 代码如下。

```
<div class="container">
    <text class="item">
        item1
    </text>
    <text class="item">
        item2
    </text>
    <text class="item">
        item3
    </text>
</div>
```

如果将上述 css 代码的 flex-direction 属性值修改为 column，则显示如图 3.3 所示的效果。

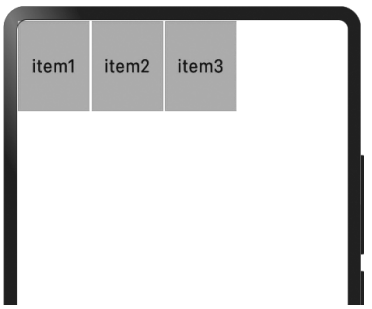

图 3.2 row 显示效果

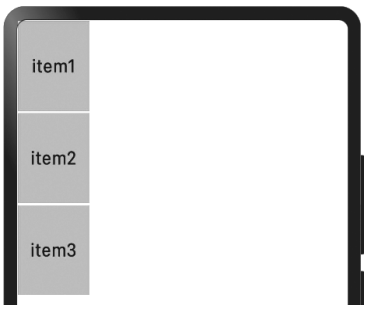

图 3.3 column 显示效果

2. flex-wrap 属性

flex-wrap 属性用于设置容器中的项目是否换行，它的属性值包含 nowrap 和 wrap。

（1）nowrap：不允许项目换行，默认值。如果所有项目按主轴方向排列时超过了容器的宽度或高度，则每个项目的宽度或高度会自动沿主轴方向压缩；如果压缩后仍然超过容器的宽度或高度，则超过的项目不显示。例如，在范例 3-5 的 hml 代码的第 10 行下再增加 3 个显示 item * 的 text 组件，其显示效果如图 3.4 所示。

（2）wrap：允许项目换行。如果所有项目按主轴方向排列时超过了容器的宽度或高度，则超过容器宽度或高度的项目会另起一行或一列。例如，在范例 3-5 的 css 代码中增加“flex-wrap：wrap；”代码，在 hml 代码的第 10 行下再增加 3 个显示 item * 的 text 组件，其显示效果如图 3.5 所示。

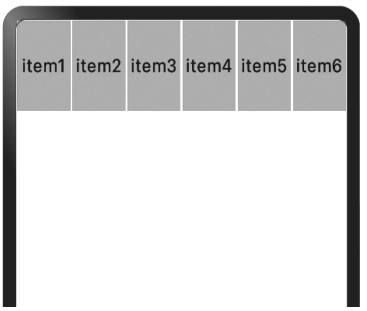

图 3.4　nowrap 显示效果

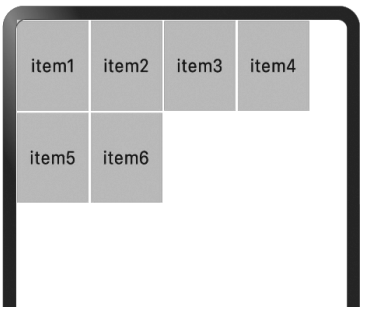

图 3.5　wrap 显示效果

3. justify-content 属性

justify-content 属性用于设置项目在主轴方向的对齐方式及分配项目与项目之间、多余空间的间隙，它的属性值包含 flex-start、center、flex-end、space-between、space-around 和 space-evenly。

（1）flex-start：默认值，项目与主轴的起点对齐，项目间不留间隙。如果主轴为水平方向，则容器中的项目左对齐；如果主轴为垂直方向，则容器中的项目顶端对齐。

（2）center：项目在主轴方向居中对齐，项目间不留间隙。例如，在范例 3-5 的 css 代码中增加“justify-content：center；”代码，其显示效果如图 3.6 所示。

（3）flex-end：项目与主轴的终点对齐，项目间不留间隙。如果主轴为水平方向，则容器中的项目右对齐；如果主轴为垂直方向，则容器中的项目底端对齐。例如，在范例 3-5 的 css 代码中增加“fjustify-content：flex-end；”代码，其显示效果如图 3.7 所示。

（4）space-between：项目沿主轴方向均匀分布，两端的项目与容器的起点、终点对齐，并且项目间的间隙相等。例如，在范例 3-5 的 css 代码中增加“justify-content：space-between；”代码，其显示效果如图 3.8 所示。

（5）space-around：项目沿主轴方向均匀分布，两端的项目与容器的起点、终点的间隙是项目与项目之间间隙的一半。例如，在范例 3-5 的 css 代码中增加“justify-content：

space-around;”代码，其显示效果如图 3.9 所示。

图 3.6　center 显示效果

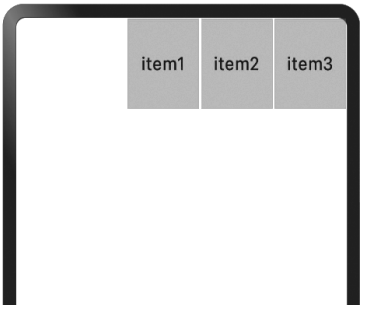

图 3.7　flex-end 显示效果

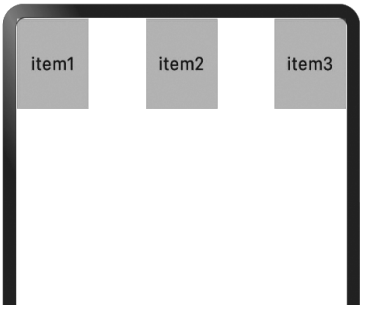

图 3.8　space-between 显示效果

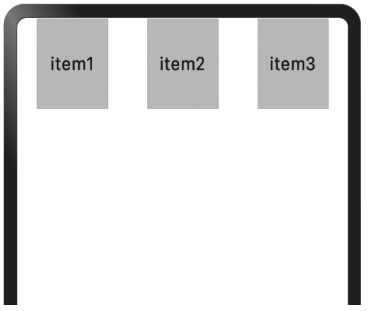

图 3.9　space-around 显示效果

(6) space-evenly：项目沿主轴方向均匀分布，每个项目之间的间隔相等。

4. align-items 属性

align-items 属性用于设置项目在交叉轴方向的对齐方式，它包含 stretch、flex-start、center、flex-end 和 baseline 5 个属性值。

(1) stretch：项目在交叉轴方向被拉伸到与容器相同的高度或宽度。

(2) flex-start：项目与交叉轴的起点对齐。

(3) center：项目在交叉轴方向居中对齐。

(4) flex-end：项目与交叉轴的终点对齐。

(5) baseline：项目与基线对齐。如果主轴为垂直方向，则该值与 flex-start 等效；如果主轴为水平方向，则项目上的文本内容按文本基线对齐，否则底部对齐。

扫一扫

3.2.2　项目的属性

1. flex-grow 属性

flex-grow 属性用于设置项目在容器主轴方向上剩余空间的拉升比例，默认值为 0(表示项目不拉升)。如果所有项目的 flex-grow 属性值相等，则它们将等分剩余空间。该属性

仅对 div、list-item、tabs、refresh、stepper-item 等容器组件生效。例如，将范例 3-5 的 hml 代码修改为如下代码，其显示效果如图 3.10 所示。

```
<div class="container">
    <text class="item" style="flex-grow : 1;">
        item1
    </text>
    <text class="item" style="flex-grow : 2;">
        item2
    </text>
    <text class="item" style="flex-grow : 1;">
        item3
    </text>
</div>
```

上述代码表示按 1∶2∶1 的比例分配剩余空间显示 item1、item2 和 item3 这三个 text 组件，即 item1 拉升四分之一剩余空间，item2 拉升四分之二剩余空间，item3 拉升四分之一剩余空间。

2. flex-shrink 属性

flex-shrink 属性用于设置项目在容器主轴方向上的收缩比例，但只有当项目的默认宽度之和大于容器的宽度时才会发生收缩，默认值为 1（表示等比收缩项目）。如果 flex-shrink 属性值为 0，表示项目不收缩。该属性仅对 div、list-item、tabs、refresh、stepper-item 等容器组件生效。

3. flex-basis 属性

flex-basis 属性用于设置项目在主轴方向上的初始宽度或高度值。如果 width 或 height 与 flex-basis 属性同时设置时，项目初始宽度或高度由 flex-basis 属性决定。该属性仅对 div、list-item、tabs、refresh、stepper-item 等容器组件生效。例如，将范例 3-5 的 hml 代码修改为如下代码，其显示效果如图 3.11 所示。

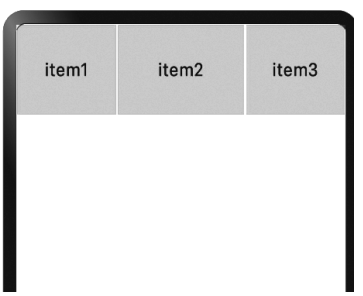

图 3.10　flex-grow 显示效果

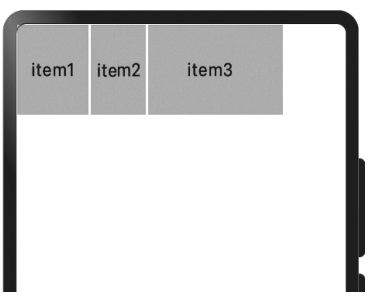

图 3.11　flex-basis 显示效果

```
<div class="container">
    <text class="item" >
```

```
        item1
    </text>
    <text class="item" style="flex-basis: 30px;">
        item2
    </text>
    <text class="item" style="flex-basis: 180px;">
        item3
    </text>
</div>
```

上述代码表示显示 item1 的 text 组件大小由 width 属性值决定，显示 item2 和 item3 的 text 组件大小由 flex-basis 属性值决定。

4. align-self 属性

align-self 属性用于设置项目在容器交叉轴上的对齐方式，该属性会覆盖容器的 align-items 属性，仅当容器为 div、list 组件时生效。它包含 stretch、flex-start、flex-end、center 和 baseline 5 个属性值，其功能含义与 align-items 属性一样，限于篇幅，这里不再赘述。

5. position 属性

position 属性用于设置项目的定位类型，不支持动态变更。它的属性值包含 fixed、absolute 和 relative，但 absolute 属性仅当容器为 div、stack 组件时生效。项目在页面上用 top 属性（离容器顶部的距离）、bottom 属性（离容器底部的距离）、left 属性（离容器左侧的距离）和 right 属性（离容器右侧的距离）进行定位，但是，只有设置了 position 属性，它们才能生效。根据不同的 position 值，它们的工作方式也不同。

（1）fixed（固定定位）：相对于整个界面窗口进行固定定位，即项目按照设置的 top、right、bottom 和 left 属性值进行固定定位，即使是滚动页面，项目也始终位于同一位置，其余项目会根据该项目通常应放置位置上留出的空隙进行调整来适应窗口尺寸。例如，将某个项目放在界面窗口右下角的 css 代码如下。

```
.info_position{
  position: fixed;
  bottom: 0px;                                    /* 离窗口底部距离为 0 */
  right: 0px;                                     /* 离窗口右侧距离为 0 */
}
```

（2）absolute（绝对定位）：相对于父容器进行定位，即设置项目的 top、right、bottom 和 left 属性值相对于最近的父容器进行定位，而不是相对于界面容器定位，其余项目不会根据该项目通常应放置位置上留出的空隙进行调整来适应窗口尺寸。如果绝对定位的项目没有祖先容器对象，则它会随页面滚动一起移动。

（3）relative（相对定位）：相对于其正常位置进行定位，即设置项目的 top、right、bottom 和 left 属性值将导致项目偏离其正常位置进行调整，其余项目不会根据该项目通常应放置位置上留出的空隙进行调整来适应窗口尺寸。

6. z-index 属性

z-index 属性用于设置同一父节点中子节点的渲染顺序，该属性值越大，渲染效果越靠后。

本章小结

用户界面是系统和用户之间进行交互和信息交换的媒介，好的界面设计不仅能让应用程序变得有个性、有品位，还能让应用程序的操作变得舒适、简单，且能充分体现应用程序的定位和特点。本章首先详细介绍了尺寸单位、通用样式、动画样式及渐变样式的定义和使用方法，然后结合实际案例阐述了 flex 布局中容器属性和项目属性的使用方法，以便读者设计符合要求的用户界面。