样式和主题背景

借助 Android 上的样式和主题背景，您可将应用设计的细节与界面的结构和行为分开，类似于网页设计中的样式表。

样式是一个属性集合，用于指定单个 `View` (/reference/android/view/View) 的外观。 样式可以指定字体颜色、字号、背景颜色等属性。

主题背景是一种应用于整个应用、Activity 或视图层次结构的样式，而不仅仅应用于单个视图。当您将样式作为主题背景来应用时，应用或 Activity 中的每个视图都会应用其支持的每个样式属性。主题背景还可以将样式应用于非视图元素，例如状态栏和窗口背景。

样式和主题背景在 `res/values/` 中的样式资源文件 (/guide/topics/resources/style-resource)中声明，通常命名为 `styles.xml`。

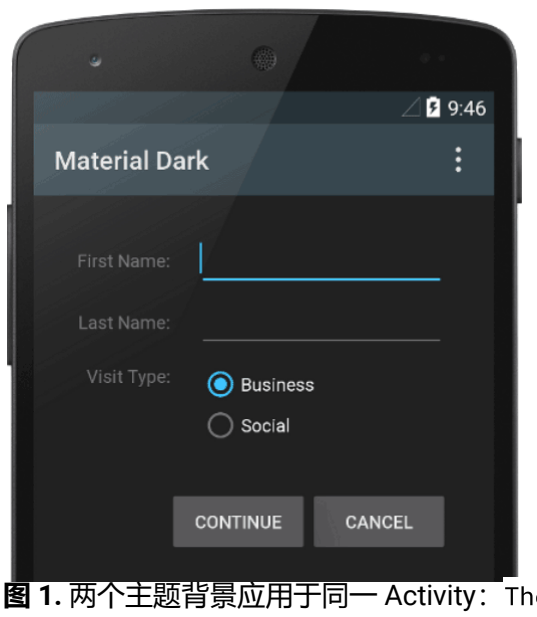

图 1. 两个主题背景应用于同一 Activity：`Theme.AppCompat`（左）和 `Theme.AppCompat.Light`（右）

创建并应用样式

要创建新样式或主题背景，请打开项目的 `res/values/styles.xml` 文件。对于您想创建的每种样式，请按以下步骤操作：

1. 使用唯一标识样式的名称添加 `<style>` 元素。
2. 为您要定义的每个样式属性添加 `<item>` 元素。

   每一项中的 `name` 会指定您原本会在布局中作为 XML 属性来使用的属性。`<item>` 元素中的值即为该属性的值。

例如，如果您定义以下样式：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

您可以将样式应用到视图，如下所示：

```
<TextView
    style="@style/GreenText"
    ... />
```

只要视图接受，样式中指定的每个属性都会应用于该视图。视图只会忽略其不接受的任何属性。

注意：只有添加了 `style` 属性的元素才会收到这些样式属性，任何子视图都不会应用这些样式。如果希望子视图继承样式，则应该改为应用具有 `android:theme` 属性的样式。

不过，您通常不会将样式应用于各个视图，而是将样式作为主题背景应用 (#Theme)于整个应用、Activity 或视图集合。

扩展和自定义样式

创建自己的样式时，应始终扩展框架或支持库中的现有样式，以保持与平台界面样式的兼容性。要扩展样式，请使用 `parent` 属性指定要扩展的样式。然后，您可以替换继承的样式属性并添加新属性。

例如，您可以继承 Android 平台的默认文本外观，并按如下所示进行修改：

```
<style name="GreenText" parent="@android:style/TextAppearance">
    <item name="android:textColor">#00FF00</item>
</style>
```

不过，您应始终继承 Android 支持库中的核心应用样式。为提供与 Android 4.0（API 级别 14）及更高版本的兼容性，支持库中的样式会针对每个版本中可用的界面属性优化各个样式。支持库样式的名称通常与平台样式相似，但包含 `AppCompat`。

要从库或您自己的项目继承样式，请声明父样式名称，但不带上方所示的 `@android:style/` 部分。 例如，以下示例继承了支持库中的文本外观样式：

```
<style name="GreenText" parent="TextAppearance.AppCompat">
    <item name="android:textColor">#00FF00</item>
</style>
```

要继承样式（平台中的样式除外），您也可以使用点符号来扩展样式名称，而不是使用 `parent` 属性。也就是说，为您的样式名称添加您想继承的样式名称作为前缀，用句点分隔。这种方式通常只能用来扩展您自己的样式，而不能用来扩展其他库的样式。例如，以下样式从上方的 `GreenText` 样式继承了所有样式，然后增加了文本大小：

```
<style name="GreenText.Large">
    <item name="android:textSize">22dp</item>
</style>
```

通过链接更多名称，您可以根据需要继续像这样继承多次样式。

注意：如果使用点符号来扩展样式，并且还包含了 `parent` 属性，则父样式将替换通过点符号继承的任何样式。

要查找可使用 `<item>` 标记声明的属性，请参阅各种类应用中的“XML 属性”表格。所有视图都支持基础 `View` 类中的 XML 属性 (/reference/android/view/View#lattrs)，而且许多数据视图都添加了自己的特殊属性。例如，TextView XML 属性 (/reference/android/widget/TextView#lattrs)包括 `android:inputType` (/reference/android/widget/TextView#attr_android:inputType) 属性，您可以将其应用于接收输入内容的文本视图，例如 `EditText` (/reference/android/widget/EditText) 微件。

将样式作为主题背景来应用

您可以像创建样式一样创建主题背景。不同之处在于应用主题背景的方式：您不是对视图应用具有 `style` 属性的样式，而不是对 `AndroidManifest.xml` 文件中的 `<application>` 标签或 `<activity>` 标签应用具有 `android:theme` 属性的主题背景。

例如，下面演示了如何将 Android 支持库的 Material Design“深色”主题背景应用于整个应用：

```
<manifest ... >
    <application android:theme="@style/Theme.AppCompat" ... >
    </application>
</manifest>
```

下面演示了如何将“浅色”主题背景仅应用于一个 Activity：

```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.AppCompat.Light" ... >
        </activity>
    </application>
</manifest>
```

现在，应用或 Activity 中的每个视图都会应用指定主题背景中定义的样式。如果视图仅支持在样式中声明的某些属性，则它仅会应用这些属性，而忽略其不支持的属性。

从 Android 5.0（API 级别 21）和 Android 支持库 v22.1 开始，您还可以在布局文件中为视图指定 `android:theme` 属性。这会修改该视图及任何子视图的主题背景，适用于更改界面特定部分中的主题背景调色板。

前面的示例展示了如何应用主题背景，例如 Android 支持库提供的 `Theme.AppCompat`。但是，您经常需要自定义主题背景来适合应用的品牌。要实现这一目的，最好的方式是对支持库中的这些样式进行扩展，并替换一些属性，如下一部分所述。

样式层次结构

Android 提供了多种在整个 Android 应用中设置属性的方法。例如，您可以直接在布局中设置属性，将样式应用到视图，将主题背景应用到布局，甚至以编程方式设置属性。

在选择如何为应用设置样式时，请注意 Android 的样式层次结构。一般来说，您应该尽量使用主题背景和样式，以保持一致性。如果您在多个位置指定了相同的属性，下面的列表决定了最终将应用哪些属性。该列表从最高优先级到最低优先级排序：

1. 通过文本 span 将字符或段落级样式应用到 `TextView` 派生类
2. 以编程方式应用属性
3. 将单独的属性直接应用到 View
4. 将样式应用到 View
5. 默认样式
6. 将主题背景应用于 View、Activity 或您的整个应用
7. 应用某些特定于 View 的样式，例如在 `TextView` 上设置 `TextAppearance` (/reference/com/google/android/material/resources/TextAppearance)

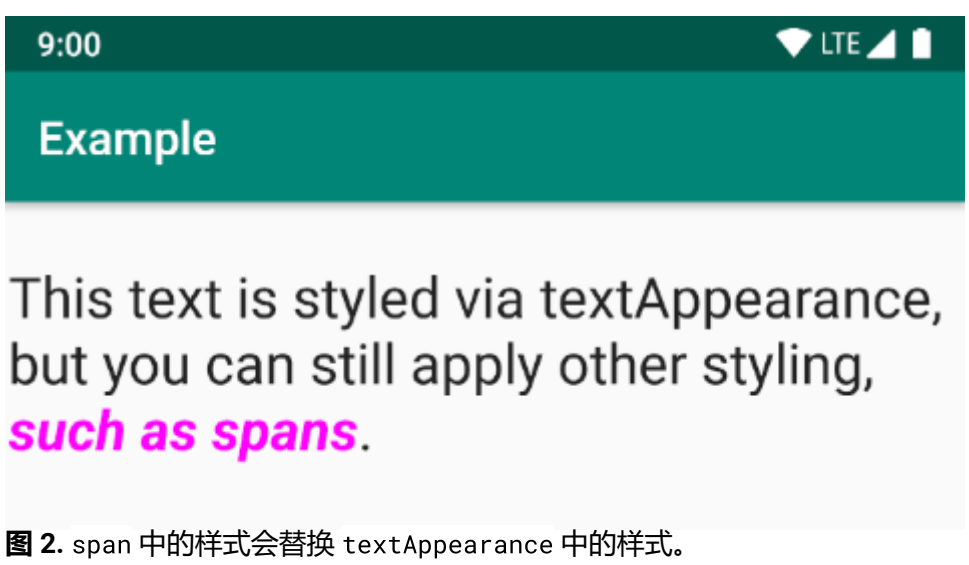

图 2. `span` 中的样式会替换 `textAppearance` 中的样式。

如果您尝试设置应用样式并且没有看到预期结果，则可能是其他样式替换了您的更改。例如，如果将主题背景应用到您的应用，同时将样式应用到单个 `View`，则样式属性会替换与该 `View` 匹配的任何主题背景属性。但请注意，未被样式替换的主题背景属性仍会使用。

TextAppearance

样式的一个局限性是您只能将一个样式应用于 `View`。但在 `TextView` 中，您也可以指定功能与样式相似的 `TextAppearance` (/reference/com/google/android/material/resources/TextAppearance) 属性，如以下示例所示：

```
<TextView
    ...
    android:textAppearance="@android:style/TextAppearance.Material.Headline"
    android:text="This text is styled via textAppearance!" />
```

`TextAppearance` 可用于定义特定于文本的样式，同时让 `View` 的样式留作他用。但是请注意，如果直接在 `View` 上或样式中定义任何文本属性，这些值会覆盖 `TextAppearance` 值。

`TextAppearance` 支持 `TextView` 提供的一部分样式属性。有关完整的属性列表，请参阅 TextAppearance (/reference/com/google/android/material/resources/TextAppearance)。

未包含的一些常见 `TextView` 属性为 `lineHeight[Multiplier|Extra]` (/reference/android/widget/TextView#attr_android:lineHeight)、`lines` (/reference/android/widget/TextView#attr_android:lines)、`breakStrategy` (/reference/android/widget/TextView#attr_android:breakStrategy) 和 `hyphenationFrequency` (/reference/android/widget/TextView#attr_android:hyphenationFrequency)。`TextAppearance` 作用于字符级别，而不作用于段落级别，因此不支持影响整个布局的属性。

自定义默认主题背景

当您使用 Android Studio 创建项目时，根据项目的 `styles.xml` 文件中的定义，它默认会将 Material Design 主题背景应用于您的应用。该 `AppTheme` 样式扩展了支持库中的主题背景，替换了应用栏 (/training/appbar)和浮动操作按钮 (/guide/topics/ui/floating-action-button)（如果使用）等关键界面元素所用的颜色属性。因此您可以通过更新提供的颜色来自定义应用的颜色设计。

例如，您的 `styles.xml` 文件应类似于下方所示：

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

请注意，样式值实际上是对项目的 `res/values/colors.xml` 文件中定义的其他颜色资源 (/guide/topics/resources/more-resources#Color)的引用。因此，要更改颜色，您应该编辑该文件。但在开始更改这些颜色之前，请使用素材颜色工具 (https://material.io/color/)预览您的颜色。您可以借助该工具从素材调色板中挑选颜色并预览它们在应用中的显示效果。

了解颜色后，请更新 `res/values/colors.xml` 中的值：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!--   color for the app bar and other primary UI elements -->
    <color name="colorPrimary">#3F51B5</color>

    <!--   a darker variant of the primary color, used for
           the status bar (on Android 5.0+) and contextual app bars -->
    <color name="colorPrimaryDark">#303F9F</color>

    <!--   a secondary color for controls like checkboxes and text fields -->
    <color name="colorAccent">#FF4081</color>
</resources>
```

然后您可以替换所需的任何其他样式。例如，您可以更改 Activity 背景颜色，如下所示：

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    ...
    <item name="android:windowBackground">@color/activityBackground</item>
</style>
```

有关您可以在主题背景中使用的属性列表，请参阅位于 `R.styleable.Theme` (/reference/android/R.styleable#Theme) 的属性表格。为布局中的视图添加样式时，您也可以查看视图类应用中的“XML 属性”表格，以查找属性。例如，所有视图都支持基础 `View` 类中的 XML 属性 (/reference/android/view/View#lattrs)。

大多数属性会应用于特定类型的数据视图，而某些属性会应用于所有视图。但是，`R.styleable.Theme` (/reference/android/R.styleable#Theme) 处列出的某些主题背景属性会应用于 Activity 窗口，而不会应用于布局中的视图。例如，`windowBackground` 可更改窗口背景，而 `windowEnterTransition` 可定义要在 Activity 启动时使用的过渡动画（如需了解详情，请参阅使用动画启动 Activity (/training/transitions/start-activity)）。

Android 支持库还提供了其他属性，可供您用来自定义从 `Theme.AppCompat` 扩展的主题背景（例如上方所示的 `colorPrimary` 属性）。最好在库的 attrs.xml 文件 (https://android.googlesource.com/platform/frameworks/support/+/master/v7/appcompat/res/values/attrs.xml)中查看这些属性

注意：支持库中的属性名称不使用 `android:` 前缀。该前缀仅用于 Android 框架中的属性。

您还可能还想扩展支持库中提供的不同主题背景，而不是上方所示的主题背景。查看可用主题背景的最佳位置是库的 themes.xml 文件 (https://android.googlesource.com/platform/frameworks/support/+/master/v7/appcompat/res/values/themes.xml)。

添加特定于版本的样式

如果新版本的 Android 添加了您想要使用的主题背景属性，您可以将它们添加到您的主题背景中，同时仍然与旧版本兼容。您只需要另一个 `styles.xml` 文件，该文件保存在包含资源版本限定符 (/guide/topics/resources/providing-resources#AlternativeResources)的 `values` 目录中。例如：

```
res/values/styles.xml        # themes for all versions
res/values-v21/styles.xml    # themes for API level 21+ only
```

由于 `values/styles.xml` 文件中的样式适用于所有版本，因此 `values-v21/styles.xml` 中的主题背景可以继承这些样式。这样，您可以先从“基础”主题背景开始，然后在特定于版本的样式中扩展该主题背景，以避免重复样式。

例如，要为 Android 5.0（API 级别 21）声明窗口转换，您需要使用一些新属性。因此您在 `res/values/styles.xml` 中的基础主题背景可能如下所示：

```
<resources>
    <!-- base set of styles that apply to all versions -->
    <style name="BaseAppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/primaryColor</item>
        <item name="colorPrimaryDark">@color/primaryTextColor</item>
        <item name="colorAccent">@color/secondaryColor</item>
    </style>

    <!-- declare the theme name that's actually applied in the manifest file -->
    <style name="AppTheme" parent="BaseAppTheme" />
</resources>
```

然后在 `res/values-v21/styles.xml` 中添加特定于版本的样式，如下所示：

```
<resources>
    <!-- extend the base theme to add styles available only with API level 21+ -->
    <style name="AppTheme" parent="BaseAppTheme">
        <item name="android:windowActivityTransitions">true</item>
        <item name="android:windowEnterTransition">@android:transition/slide_right</item>
        <item name="android:windowExitTransition">@android:transition/slide_left</item>
    </style>
</resources>
```

现在，您可以在清单文件中应用 AppTheme，系统会选择适用于每个系统版本的样式。

如需详细了解如何为不同设备使用备用资源，请参阅提供资源 (/guide/topics/resources/providing-resources)。

自定义微件样式

框架和支持库中的每个微件都有一个默认样式。例如，当您使用支持库中的主题背景为应用设置样式时，Button (/reference/android/widget/Button) 的实例会使用 Widget.AppCompat.Button (/reference/androidx/appcompat/R.style#Widget_AppCompat_Button) 样式来设置样式。如果您希望将不同的微件样式应用于某个按钮，则可以使用布局文件中的 style 属性来执行此操作。例如，以下代码会应用库的无边框按钮样式：

```
<Button
    style="@style/Widget.AppCompat.Button.Borderless"
    ... />
```

如果您想将此样式应用于所有按钮，您可以在主题背景的 buttonStyle (/reference/android/R.attr#buttonStyle) 中声明该样式，如下所示：

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="buttonStyle">@style/Widget.AppCompat.Button.Borderless</item>
    ...
</style>
```

您还可以扩展微件样式，就像扩展任何其他样式 (#Customize)一样，然后在布局或主题背景中应用自定义微件样式。

要了解支持库提供的所有备用微件样式，请查看以 Widget 开头的字段的 R.style (/reference/androidx/appcompat/R.style) 参考信息。（忽略以 Base_Widget 开头的样式。）在资源中使用样式名称时，请记得将所有下划线替换为句点。

Last updated 2019-12-30 UTC.