Chapter 11
Overview

- Learn how to style your application with support libraries for applying themes and styles
- Learn how to define a circle shape drawable
- Learn about theme and style Inheritance
- Learn how to applying merge and include to your layouts
- Learn about new layouts such as TextInputLayout, FloatingActionButton, and using the Toolbar as a bottom bar
- Learn how to apply application branding
One aspect of Android application development that is frequently overlooked is styles and themes.

With a little bit of effort, the most boring-looking application can be transformed by applying styles and themes.

Understanding how styles and themes work is important.

It is easy to make using styles and themes overly complex when it actually is fairly simple.

An example will be presented showing the default Android styling applied to a basic layout, followed by applying colors, styles, and themes to that same layout.

You will see how performing a few actions could change the entire feel of your application.

By the end of the chapter, you should be able to apply these concepts to your own application with ease.
Styling with Support

- In this chapter, you will make use of two Android support libraries.
  - To learn about adding support libraries to your application, see Appendix E, “Quick-Start: Gradle Build System.”
- We will be adding the appcompat-v7 support library and the design support library as Gradle module dependencies.
- To do so, add the following two lines to the dependencies section of the build.gradle app module file:
  ```java
  compile 'com.android.support:appcompat-v7:23.0.0'
  compile 'com.android.support:design:23.0.0'
  ```
- In addition, your application activities should extend the AppCompatActivity support class rather than Activity.
Although many people confuse themes and styles, they are actually two very different things.

- A theme is applied to an application or Activity as a whole.
  - It is usually best suited for the branding of your application.
- A style is usually applied to one particular View, rather than an entire application or Activity.
Themes and Styles (Cont’d)

- For example, if you set the color of a particular attribute of a theme via `android:.textColor`, whatever color you choose will be applied to all text in the application.
- Whereas, if you apply the `textColor` attribute directly to a `TextView`, that `textColor` will only be applied to that particular `TextView`.
- Further, if you define a style that defines the `textColor` and you apply the style to a `TextView`, only the `TextView` will have that color applied.
Themes and Styles (Cont’d)

- In this chapter we will be working with both themes and styles.
- The themes reside in:
  - res/values/themes.xml and res/values-v21/themes.xml
- The styles reside in:
  - res/values/styles.xml and res/values-v21/styles.xml
Defining the Default Application Themes

- We will be placing all backward-compatible themes in the `res/values/themes.xml` file.
- For any styles for API Level 21 or newer, we will be placing them in the `res/values-v21/themes.xml` file.
- The default application theme will inherit from `Theme.AppCompat` and we will be using a `Toolbar` as the `ActionBar`, so the NoActionBar theme must be selected.
Defining the Default Application Themes (Cont’d)

<style name="Brand" parent="Theme.AppCompat.NoActionBar"/>
Defining the Default Application Themes (Cont’d)

<style name="Toolbar" parent="Theme.AppCompat"/>
Defining the Default Application Themes (Cont’d)
Defining a Circle Shape Drawable

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid
        android:color="@color/circle" />

    <size
        android:width="40dp"
        android:height="40dp"/>
</shape>
```
Theme and Style Inheritance

- Just like Java classes can inherit functionality from other classes, the same goes for themes and styles.
- This means you can apply attributes to a theme or style, and inherit a new theme or style from the one you defined as well as override any existing attributes or define new attributes.
- All themes for the StylesAndThemes application will inherit from the Brand or Toolbar theme that we applied earlier.
- There will be two themes, one Green and one Orange, but the majority of this chapter will focus on the Green theme.
Theme and Style Inheritance (Cont’d)

<style name="Brand.Green" parent="Brand">
    <!-- Define your green brand theme here -->
</style>
Theme and Style Inheritance (Cont’d)

```html
<style name="Toolbar.Green" parent="Toolbar">
    <!-- Define your green toolbar theme here -->
</style>
```
The colors for the application are defined in the `res/values/colors.xml` file.

To define a color, you simply add an RGB value to a color tag in the `colors.xml` file as follows:

```
<color name="black">#000000</color>
```

This adds the `#000000`, which is the RGB value of the color black to a color resource named `black`.

You may also reference defined color resources that you define as follows:

```
<color name="circle">@color/black</color>
```

This references the `black` color you defined with the `@color/black` value and applies that color to the `circle` color resource.

This is useful for creating a master list of colors and then referencing that color rather than using the RGB value all across your application.

If you have to change the color, all you need to do is change that color in one location.
Colors (Cont’d)

<!-- Green Activity Branding Colors -->

<table>
<thead>
<tr>
<th>Color Name</th>
<th>Color Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>theme_green_primary_dark</td>
<td>@color/light_green_900</td>
</tr>
<tr>
<td>theme_green_primary</td>
<td>@color/light_green_700</td>
</tr>
<tr>
<td>theme_green_accent</td>
<td>@color/amber_900</td>
</tr>
<tr>
<td>theme_green_background</td>
<td>@color/theme_green_primary</td>
</tr>
<tr>
<td>theme_green_control_highlight</td>
<td>@color/theme_green_primary</td>
</tr>
<tr>
<td>theme_green_status_bar</td>
<td>@color/theme_green_primary_dark</td>
</tr>
<tr>
<td>theme_green_action_bar</td>
<td>@color/theme_green_primary</td>
</tr>
<tr>
<td>theme_green_window_background</td>
<td>@color/light_green_100</td>
</tr>
<tr>
<td>theme_green_bottom_bar</td>
<td>@color/theme_green_primary</td>
</tr>
<tr>
<td>theme_green_nav_bar</td>
<td>@color/theme_green_primary_dark</td>
</tr>
<tr>
<td>theme_green_linear_content_background</td>
<td>@color/light_green_200</td>
</tr>
<tr>
<td>theme_green_heading_text_color</td>
<td>@color/amber_900</td>
</tr>
<tr>
<td>theme_green_toolbar_overflow_text_color</td>
<td>@color/light_green_100</td>
</tr>
</tbody>
</table>

<!-- Text color for Green -->

<color name="theme_green_text_color">@color/brown_700</color>
Colors (Cont’d)

```xml
<style name="Brand.Green" parent="Brand">
  <!-- android:windowBackground colors the root background area of the app -->
  <item name="android:windowBackground">@color/theme_green_window_background</item>
  <!-- colorPrimaryDark colors the status bar -->
  <item name="colorPrimaryDark">@color/theme_green_primary_dark</item>
  <!-- colorPrimary colors the action bar and toolbar -->
  <item name="colorPrimary">@color/theme_green_primary</item>
  <!-- colorAccent colors the floating action button and accents of controls -->
  <item name="colorAccent">@color/theme_greenAccent</item>
  <!-- colorControlHighlight controls the material ripple color -->
  <item name="colorControlHighlight">@color/theme_green_control_highlight</item>
  <!-- android:textColor controls the color of text in the app -->
  <item name="android:textColor">@color/theme_green_text_color</item>
  <!-- android:textColorHint controls the color of hint in the EditText -->
  <item name="android:textColorHint">@color/theme_green_primary_dark</item>
</style>
```
Colors (Cont’d)

- There are a few theme attributes you should be aware of for color branding your application.
- These attributes are useful when you have chosen a color palette for styling system attributes beyond the style color attributes:
  - `colorPrimary`
  - `colorAccent`
  - `colorPrimaryDark`
  - `colorControlHighlight`
  - `statusBarColor`
  - `navigationBarColor`
  - `android:windowBackgroundColor`
Rather than discussing every detail of how the layout has been created, we ask that you refer to the StylesAndThemes application layout files.

The `activity_styles_and_themes.xml` and the `toolbar.xml` file show the default layout with basic positioning applied without styles.

Use this file as a guide to see how this layout has been created and compare it to the Green theme.

The equivalent layout is the `activity_green_brand.xml` and the `green_toolbar.xml` files.

We will be discussing the most notable features of these layouts.
Merge and Include

- You may have noticed the toolbar.xml file.
- This file has the root XML tag of <merge> with a child Toolbar widget defined.
- The <merge> tag allows you to make reusable components for your application.
- Rather than defining the same component multiple times in a layout, you could define it once in its own file with <merge> as the root tag and then include that layout in another layout.
- You can also use something like a LinearLayout or RelativeLayout as the root of the Toolbar.
- Then include that layout in another layout.
  - The <merge> tag will not add additional resource consumption, whereas the LinearLayout or RelativeLayout will.
<merge>
  <android.support.v7.widget.Toolbar
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:background="@color/default_toolbar"
    android:elevation="@dimen/highEle"
    android:minHeight="?attr/actionBarSize"
    tools:showIn="@layout/activity_styles_and_themes" />
</merge>
Include Applied

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relative_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DefaultBrandActivity">

    <include layout="@layout/toolbar" />

</RelativeLayout>
A new widget that was added to the design support library is the TextInputLayout.

This widget wraps an EditText and allows the android:hint attribute of the EditText to float above the EditText for constant viewing, rather than the hint disappearing once a user begins entering text into the EditText field.
TextInputLayout (Cont’d)

<android.support.design.widget.TextInputLayout
    android:id="@+id/input_layout02"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_toEndOf="@id/circle05">

    <EditText
        android:id="@+id/editText02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/hint"
        android:text="@string/editText" />

</android.support.design.widget.widget.TextInputLayout>
FloatingActionButton

<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/bottom_bar"
    android:layout_alignParentEnd="true"
    android:layout_marginBottom="@dimen/mediumdp"
    android:layout_marginEnd="@dimen/mediumdp"
    android:contentDescription="@string/fab"
    android:elevation="@dimen/highEle"
    android:src="@android:drawable/ic_input_add"
    android:tint="@color/default_fab_tint" />

Toolbar as Bottom Bar

- On older versions of Android, you were able to split an ActionBar.
- This was useful if your application needed to include action items on the ActionBar where—on smaller screens—there may not be enough room for displaying all the actions you would like displayed.
- Rather than filling up the overflow menu with action items, you could choose to split the ActionBar so that it would appear at the bottom of the application for including the action items on the bottom bar.
- This same functionality can be achieved by including a Toolbar at the bottom of your application’s layout.
<android.support.v7.widget.Toolbar
    android:id="@+id/bottom_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:background="@color/default_toolbar"
    android:elevation="@dimen/midEle"
    android:minHeight="?attr/actionBarSize"
    android:theme="@style/Toolbar">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/transparent"
    android:orientation="horizontal">

...
<ImageButton android:id="@+id/map_button"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:contentDescription="@string/map"
    android:src="@android:drawable/ic_dialog_map" />
<ImageButton android:id="@+id/email_button"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:contentDescription="@string/email"
    android:src="@android:drawable/ic_dialog_email" />
<ImageButton android:id="@+id/info_button"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:contentDescription="@string/info"
    android:src="@android:drawable/ic_dialog_info" />

</LinearLayout>
</android.support.v7.widget.Toolbar>
Application Branding

<style name="Brand.Green" parent="Brand">
    <!- android:windowBackground colors the root background area of the app -->
    <item name="android:windowBackground">@color/theme_green_window_background</item>
    <!- colorPrimaryDark colors the status bar -->
    <item name="colorPrimaryDark">@color/theme_green_primary_dark</item>
    <!- colorPrimary colors the action bar and toolbar -->
    <item name="colorPrimary">@color/theme_green_primary</item>
    <!- colorAccent colors the floating action button and accents of controls -->
    <item name="colorAccent">@color/theme_green_accent</item>
    <!- colorControlHighlight controls the material ripple color -->
    <item name="colorControlHighlight">@color/theme_green_control_highlight</item>
    <!- android:textColor controls the color of text in the app -->
    <item name="android:textColor">@color/theme_green_text_color</item>
    <!- android:textColorHint controls the color of hint in the EditText -->
    <item name="android:textColorHint">@color/theme_green_primary_dark</item>
</style>
Application Branding (Cont’d)

<style name="Toolbar.Green" parent="Toolbar">
  <!-- toolbar overflow background color controled with colorBackground -->
  <item name="android:colorBackground">@color/theme_green_accent</item>
  <!-- toolbar title color controlled with textColorPrimary-->
  <item name="android:textColorPrimary">@color/theme_green_text_color</item>
  <!-- toolbar overflow icon color controlled with textColorSecondary -->
  <item name="android:textColorSecondary">@color/theme_green_text_color</item>
  <!-- toolbar overflow text color controlled with textColor -->
  <item name="android:textColor">
    @color/theme_green_toolbar_overflow_text_color</item>
</style>
Application Branding (Cont’d)

<activity
    android:name="com.introttoandroid.stylesandthemes.GreenBrandActivity"
    android:label="@string/title_activity_green_brand"
    android:theme="@style/Brand.Green"/>
Application Branding (Cont’d)

android:background="@color/theme_green_primary"
android:theme="@style/Toolbar.Green"
Dividers and Gaps

- Another technique for creating visually appealing applications is to use dividers and gaps within your content layout.
- Dividers are small 1dp wide lines that appear to create visual divisions between content.
- Gaps are larger, sometimes 8dp, and make use of what's known as a white space for creating divisions in layouts.
- The StylesAndThemes application makes use of both dividers and gaps.
Dividers and Gaps (Cont’d)

<View
    android:background="@color/layout_divider_color"
    android:layout_width="match_parent"
    android:layout_height="1dp"
    android:alpha="0.1"/>
Dividers and Gaps (Cont’d)

```xml
<View
    android:background="@android:color/transparent"
    android:layout_width="match_parent"
    android:layout_height="8dp"/>
```
Dividers and Gaps (Cont’d)

<style name="LayoutDivider">
  <item name="android:background">@color/layout_divider_color</item>
  <item name="android:layout_width">match_parent</item>
  <item name="android:layout_height">@dimen/divider</item>
  <item name="android:alpha">0.1</item>
</style>
Dividers and Gaps (Cont’d)

<View style="@style/LayoutDivider" />

Another way to provide branding to your application is to style any menus that your application displays.

Menus overlay other UI elements and so it is important that, in addition to complementary colors, any colors you apply to a menu have the proper contrast to your application’s theme.
The Results Applied
Typography

<style name="HeadingOrange">
    <item name="android:fontFamily">casual</item>
</style>

<style name="PrimaryTextViewOrange">
    <item name="android:fontFamily">casual</item>
</style>

<style name="SecondaryTextViewOrange">
    <item name="android:fontFamily">casual</item>
</style>

<style name="EditTextOrange">
    <item name="android:fontFamily">casual</item>
</style>
Typography (Cont’d)
Chapter 11
Summary

- We have learned about styling with support libraries for applying themes and styles.
- We have learned how to define a circle shape drawable.
- We have learned about theme and style inheritance.
- We have learned how to apply merge and include to your layouts.
- We have learned how to use new layouts such as TextInputLayout, FloatingActionButton, and how to use the Toolbar as a bottom bar.
- We have learned about application branding.
References and More Information

- **Android Training: "Maintaining Compatibility": "Define Alternative Styles":**  
  http://d.android.com/training/material/compatibility.html#Theme
- **Android Training: "Styling the Action Bar":**  
  http://d.android.com/training/basics/actionbar/styling.html
- **Android Training: "Supporting Different Platform Versions": "Use Platform Styles and Themes":**  
  http://d.android.com/training/basics/supporting-devices/platforms.html#style-themes
- **Android Training: "Using the Material Theme": "Customizing the Color Palette":**  
  http://d.android.com/training/material/theme.html#ColorPalette
- **Android Training: "Re-using Layouts with <include/>":**  
  http://d.android.com/training/improving-layouts/reusing-layouts.html
- **Google Design Spec: "Style": "Color":**  
  http://www.google.com/design/spec/style/color.html
- **Google Design Spec: "Style": "Imagery":**  
  http://www.google.com/design/spec/style/imagery.html
References and More Information

- Google Design Spec: "Style": "Typography":
  - http://www.google.com/design/spec/style/typography.html

- Android API Guides: "Buttons": "Styling Your Button":
  - http://d.android.com/guide/topics/ui/controls/button.html#Style

- Android API Guides: “Menus”:

- Android API Guides: “Styles and Themes”:

- Android API Guides: “Styles Resource”:
  - http://d.android.com/guide/topics/resources/style-resource.html

- Android SDK Reference regarding the application R.style class:

- Android SDK Reference regarding the application android.text.style package:

- Android SDK Reference regarding the application Resources.Theme class:
  - http://d.android.com/reference/android/content/res/Resources.Theme.html