What is a style in android?

- A collection of properties that specify the look and format for a view
- Can specify properties such as height, padding, font color, font size, background color, etc.
- Is defined in an XML resource that is separate from the XML that specifies the layout

Styles in Android are like CSS in web design

```html
<p>
This is a paragraph
</p>

p {
  color: red;
  text-align: center;
}
```
An example for applying a style

• This is a text view **before** applying a style:

  ```xml
  <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textColor="#00FF00"
      android:typeface="monospace"
      android:text="@string/hello" />
  ```

• And here it is afterwards:

  ```xml
  <TextView
      style="@style/CodeFont"
      android:text="@string/hello" />
  ```

What is a theme?

• **A theme** is a style applied to an entire Activity or application rather than an individual View.

• When a style is applied as a theme, every View in the Activity or application will apply each style property that it supports.

• For example, you can apply the style **CodeFont** as a theme for an Activity and then all text inside that Activity will have green monospace font.

  ```xml
  AndroidManifest.xml – applying a style to the entire application
  <application android:theme="@style/CustomTheme"/>
  ```

  ```xml
  AndroidManifest.xml – applying a style to a specific activity
  <activity android:theme="@android:style/Theme.Dialog"/>
  ```
How to create styles?

• Create an XML file `res/values/styles.xml`
• Create a `<resources>` root element
• Create a `<style>` element, with `<item>` elements describing the properties of the style
  • Many `<style>` elements may be defined

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textColor">#00FF00</item>
    <item name="android:typeface">monospace</item>
  </style>
</resources>
```

• Each style element is converted into a resource object and can be referenced from an XML layout via `@style`
• Remember that a style may also be used as a theme

Style inheritance

• Use the `parent` attribute to inherit properties from built-in platform styles

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

• To inherit from your own defined style (e.g., from `CodeFont`), you may use this naming style instead:

```xml
<style name="CodeFont.Red">
  <item name="android:textColor">#FF0000</item>
</style>
```
What kind of style properties are available?

- See the Javadoc of the specific view for the list of supported XML attributes
- Note that basically all attributes may be defined in a style, for example:

```xml
<EditText
    android:inputType="number"
    ... />

<style name="Numbers">
    <item name="android:inputType">number</item>
    ... 
</style>

Note: do not forget to prefix the attribute name with the android: namespace

Some of the properties are for theme only

- See the R.attr reference that lists all possible XML attributes
- Attributes that begin with ‘window’ are effective only when the style is applied as a theme, e.g.,
  - windowNoTitle, windowBackground

Note: when applying a style to a view, the view will apply only those properties that it supports and will ignore others
How to use newer themes while being backward compatible?

- Define a custom theme that uses resource selection based on platform version
  
  ```xml
  <style name="LightThemeSelector" parent="android:Theme.Holo.Light">
  ...
  </style>
  ```

- For example, this holographic theme will be used by applications running android 3.0+
  
  ```xml
  <style name="LightThemeSelector" parent="android:Theme.Holo.Light">
  ...
  </style>
  ```

- While a default light theme will be used for older versions:
  
  ```xml
  <style name="LightThemeSelector" parent="android:Theme.Light">
  ...
  </style>
  ```

Android’s Material Design was introduced in Android 5.0 (API level 21), hence the folder `res/values-v21` may be in use.

Using platform styles and themes

- Android platform provides a large collection of styles and themes that you can use
- All available styles can be found in `R.style` class
- To use those styles, replace each underscore with a period, e.g., `Theme_NoTitleBar` => "@android:style/Theme.NoTitleBar"

- To understand the properties of each provided style, see `styles.xml`, and `themes.xml`

  For example, what properties are used to style `Theme.Dialog`?
Material Design

An Introduction

http://developer.android.com/training/material/index.html

Do you want your app to look that way?

https://www.youtube.com/watch?v=Q8TXgCzxEnw
What is material design?

• “A comprehensive guide for visual, motion, and interaction design across platforms and devices”
• Detailed design guidelines are described in the material design specification: https://material.io/guidelines/

Note: using material design requires Android 5.0 (API level 21), however many features are supported on earlier versions. See Maintaining Compatibility.

How to apply material design to my app?

• Specify a style that inherits from android:Theme.Material
• Your layouts and widgets should conform to the material design guidelines, including
  • Specifying elevation for views
  • Specialized widgets for lists and cards
  • Customized animations
  • Touch feedback

<res/values/styles.xml -->
<resources>
  <!-- your theme inherits from the material theme -->
  <style name="AppTheme" parent="android:Theme.Material">
  <!-- theme customizations -->
  </style>
</resources>
Material theme – easy color customization (1)

• Material design themes allow easy customization of the app’s color palette
• You should inherit from a base material theme and define colors that fit your brand, as follows:

```xml
<resources>
    <!-- inherit from the material theme -->
    <style name="AppTheme" parent="android:Theme.Material">
        <!-- Main theme colors -->
        <item name="android:colorPrimary">@color/primary</item>
        <!-- darker variant for the status bar and contextual app bars -->
        <item name="android:colorPrimaryDark">@color/primary_dark</item>
        <!-- theme UI controls like checkboxes and text fields -->
        <item name="android:colorAccent">@color/accent</item>
    </style>
</resources>
```

See more material themes in `R.style`

Colors may be defined in `values/colors.xml`, e.g.,

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="menu_background">#666666</color>
</resources>
```

Material theme – easy color customization (2)

• Note that the color of a variety of screen elements may be customized, even the status bar at the top
Cards

• “A card is a sheet of material that serves as an entry point to more detailed information. A card could contain a photo, text, and a link about a single subject.”
  https://www.google.com/design/spec/components/cards.html#

  - Cards are implemented using CardView widgets.
  - Use card_view:cardElevation attribute for creating a card shadow
  - See more details here

Lists

• Use the RecyclerView widget to create a complex list in your app

  The RecyclerView widget is a more advanced and flexible version of ListView
  • Flexible layout managers to position elements within the list
  • Animations for adding and removing items

• See more details here
Elevation and shadows

• Material design introduces elevation for UI elements
• The elevation of a view is represented by the Z property and determines the visual appearance of its shadow
• Elevation is also useful to create animations where widgets temporarily rise above the view plane when performing some action

Key attributes are android:elevation and android:translationZ
android:elevation refers to the elevation’s static component
android:translationZ refers to the dynamic components used for animations

Customizing view shadows

• The bounds of a view’s background drawable determine the default shape of its shadow

```xml
<shape xmlns:android="http://schemas.android.com/apk/res/android"
      android:shape="rectangle">
  <solid android:color="#42000000" />
  <corners android:radius="5dp" />
</shape>
```

- The view will have a shadow with rounded corners, since we specified a radius attribute
- Providing a custom outline overrides the default shape of a view’s shadow
Defining custom animations

• The material theme provides some default animations for buttons and activity transitions
• Android 5.0 (API level 21) and above lets you customize these animations and create new ones such as
  • Touch feedback
  • Activity transitions
  • Reveal animations (show/hide)
• See this guide for more details and code examples