Customizing App Appearance with Styles and Themes

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

What to Expect from This Module

- The Role of Styles
- Declaring and Using Styles
- The Role of Themes
- Modifying Themes with Theme Editor
- Setting Default View Styling with a Theme

View Styles

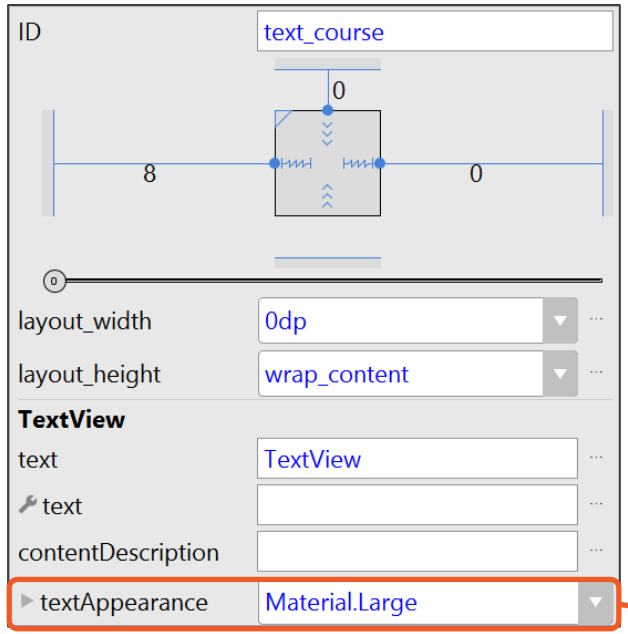
View appearance controlled by properties

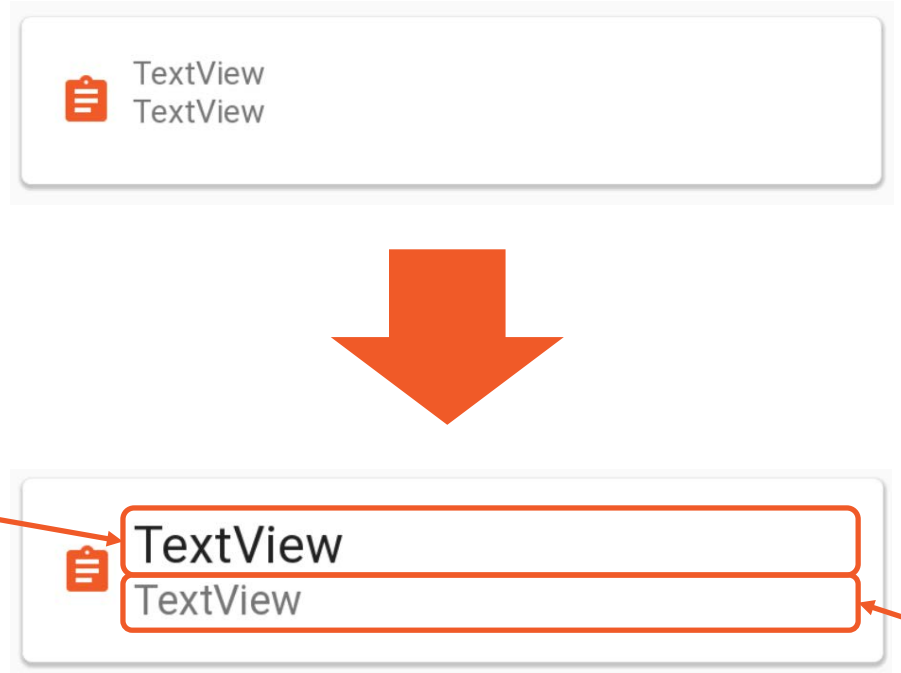
- Properties identify appearance attributes
- Often set on each view individually

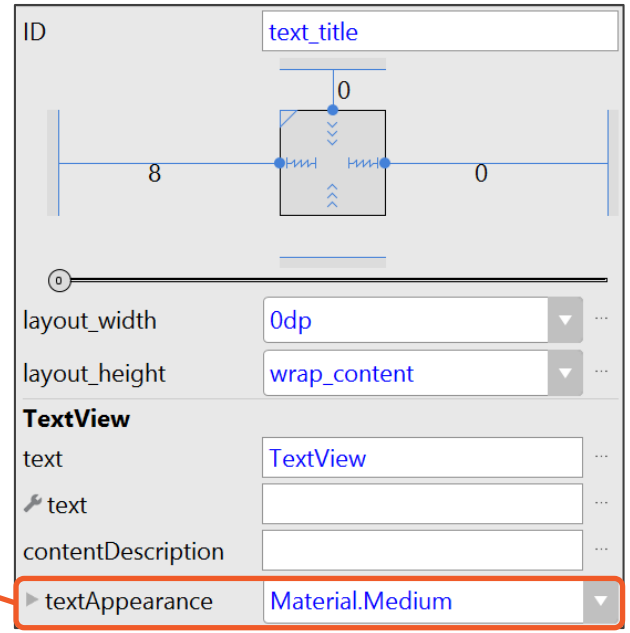
Styles define a collection of attributes

- Define and name attribute collection
- Associated with specific views
  - View takes on attributes in collection

View Styles


ID
text_course
0
8
0
layout_width
0dp
layout_height
wrap_content
TextView
text
TextView
text
contentDescription
textAppearance
Material.Large


TextView
TextView
TextView
TextView


ID
text_title
0
8
0
layout_width
0dp
layout_height
wrap_content
TextView
text
TextView
text
contentDescription
textAppearance
Material.Medium

Declaring View Styles

Styles declared as a values resource

- Declared with style element
- Child elements provide style attributes
  - Each declared with item element
  - Identified with name

Declaring View Styles

```
<resources>
    <style name="myTextStyle">
      <item name="android:textSize">20sp</item>
      <item name="android:textColor">#000000</item>
    </style>
</resources>
```

Declaring View Styles

Each view supports specific style attributes

- Provided in view class' documentation
- See the "XML attributes" section

TextView class' XML attributes

XML attributes	
android:autoLink	Controls whether links such as urls and email addresses are automatically found and converted to clickable links.
android:autoSizeMaxTextSize	The maximum text size constraint to be used when auto-sizing text.
android:autoSizeMinTextSize	The minimum text size constraint to be used when auto-sizing text.
android:autoSizePresetSizes	Resource array of dimensions to be used in conjunction with autoSizeTextType set to uniform.
android:autoSizeStepGranularity	Specify the auto-size step size if autoSizeTextType is set to uniform.
android:autoSizeTextType	Specify the type of auto-size.
android:autoText	If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors.
android:breakStrategy	Break strategy (control over paragraph layout).
android:bufferType	Determines the minimum type that getText() will return.
android:capitalize	If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types.
android:cursorVisible	Makes the cursor visible (the default) or invisible.
android:digits	If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept.
android:drawableBottom	The drawable to be drawn below the text.
android:drawableEnd	The drawable to be drawn to the end of the text.

Declaring View Styles

Style declaration can inherit another style

- Can inherit framework or project styles
- Include parent on style element
- New style will have parent attributes
- Can add/override attributes

Declaring View Styles

```
<resources>
    <style name="myBrandedTextStyle" parent="myTextStyle">
      <item name="android:textColor">#f05a28</item>
      <item name="android:textAllCaps">true</item>
    </style>
</resources>
```

Declaring View Styles

Shortcut syntax for inheriting a project style

- Can use style naming to indicate parent

Declaring View Styles

```
<resources>
    <style name="myBrandedTextStyle" parent="myTextStyle">
      <item name="android:textColor">#f05a28</item>
      <item name="android:textAllCaps">true</item>
    </style>
</resources>
```

Declaring View Styles

```
<resources>
    <style name="myTextStyle.Branded">
      <item name="android:textColor">#f05a28</item>
      <item name="android:textAllCaps">true</item>
    </style>
</resources>
```

Applying a Style to A View

Set view's style property to style name

- Applies supported attributes
- Ignores any unsupported attributes

Some view's have additional properties

- Apply a specific subset of attributes
- Example:
  - TextView's textAppearance property

Style only applies to the specific view

- Does not affect child/descendant views

Themes

Broadly applied styles

- Can be applied at the activity level
- Can be applied at application level

Applying theme to activity

- Affects the activity
- Affects the views within the activity

Applying theme to application

- Defines default theme for app activities

Themes

Applying a theme

- Set in application manifest
- Setting a specific activity's theme
  - Use theme attribute activity
- Setting the default theme
  - Use theme attribute of application

Themes

Themes defined as a style resource

- Usually inherit from an existing theme

Setting theme attributes

- Can edit style resource directly
- Can use Android Studio Theme Editor
  - Provides UI for commonly modified theme attributes
  - Provides preview of theme effects

Summary

Styles

- Define a named attribute collection
- Provide alternative to setting property values individually on each view

Summary

Declaring styles

- Declared as a values resource
- Use style element
- Contain item elements as children
  - Specify attribute values

A style can inherit another style

- Specifying parent in style declaration
- Short-hand for inheriting project styles
  - Prefix name with parent name & dot
- Can add/override parent attributes

Summary

Themes

- Broadly applied styles
- Can be applied to activity or app

Applying theme to activity

- Affects the activity
- Affects the views within the activity

Applying theme to application

- Defines default theme for app activities

Summary

Themes defined as a style resource

- Editable with Theme Editor
- Can edit resource directly

Theme attributes

- Can contain individual attribute values
- Can define default styling for views