

Level/Subject : **D3/Programming 6**
Topic : Style and Themes
Week : 9th
Activity : Creating Android apps with style & theme in Android Studio
Alocated time : 120 mins labs
Deliverables : Project folder
Due date : end of week

Competency :

Student expected to be able to create Android apps with style & theme using Android Studio IDE.

Example Practice Task:

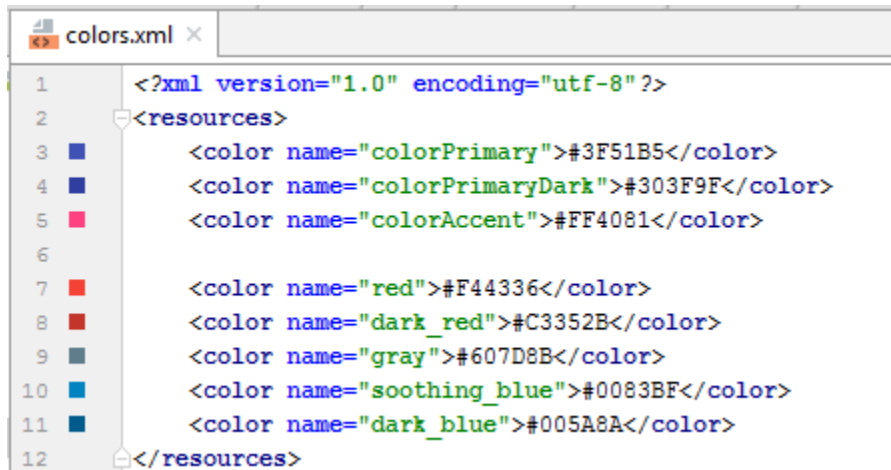
Create an Android apps with style & theme using Android Studio.

1. So far, ScreamBox sticks with the default UI styles. The buttons are stock. The colors are stock. The app does not stand out. It does not have its own brand.
2. We can restyle it.



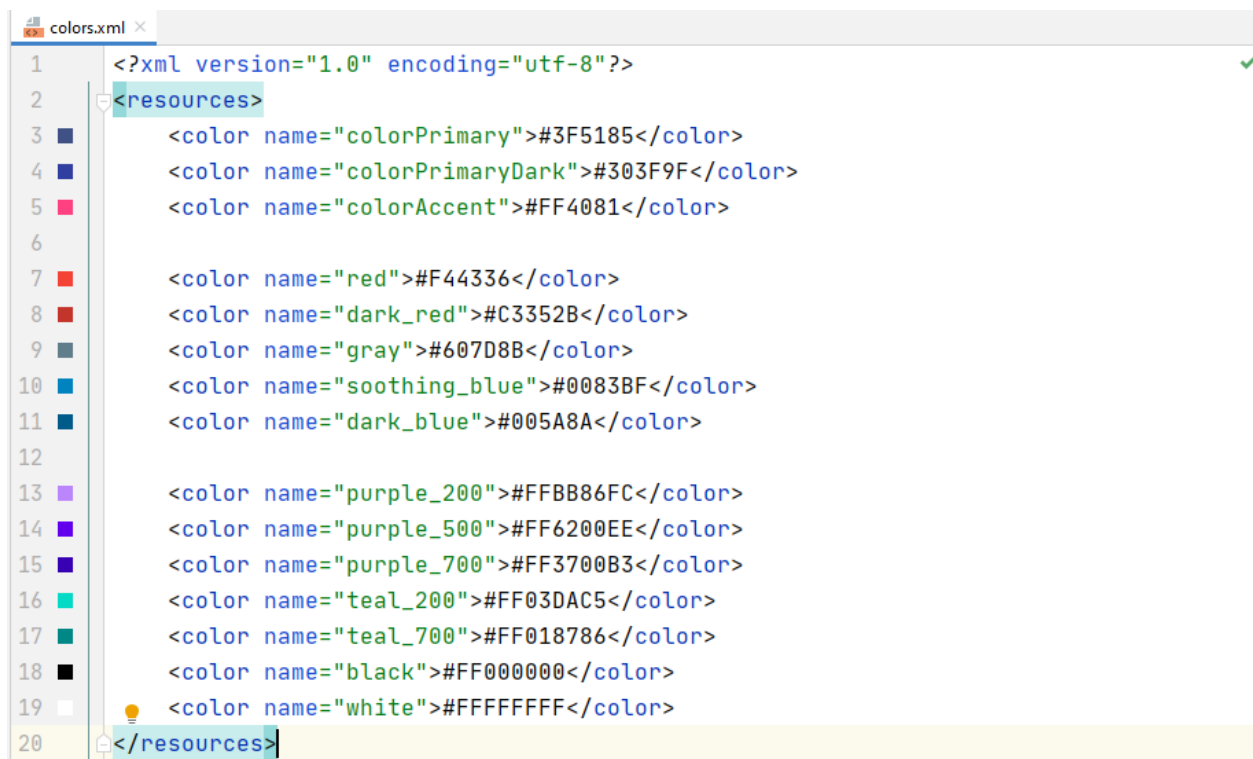
Color Resources

3. Edit your colors.xml file in res/values



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="colorPrimary">#3F51B5</color>
4     <color name="colorPrimaryDark">#303F9F</color>
5     <color name="colorAccent">#FF4081</color>
6
7     <color name="red">#F44336</color>
8     <color name="dark_red">#C3352B</color>
9     <color name="gray">#607D8B</color>
10    <color name="soothing_blue">#0083BF</color>
11    <color name="dark_blue">#005A8A</color>
12 </resources>
```

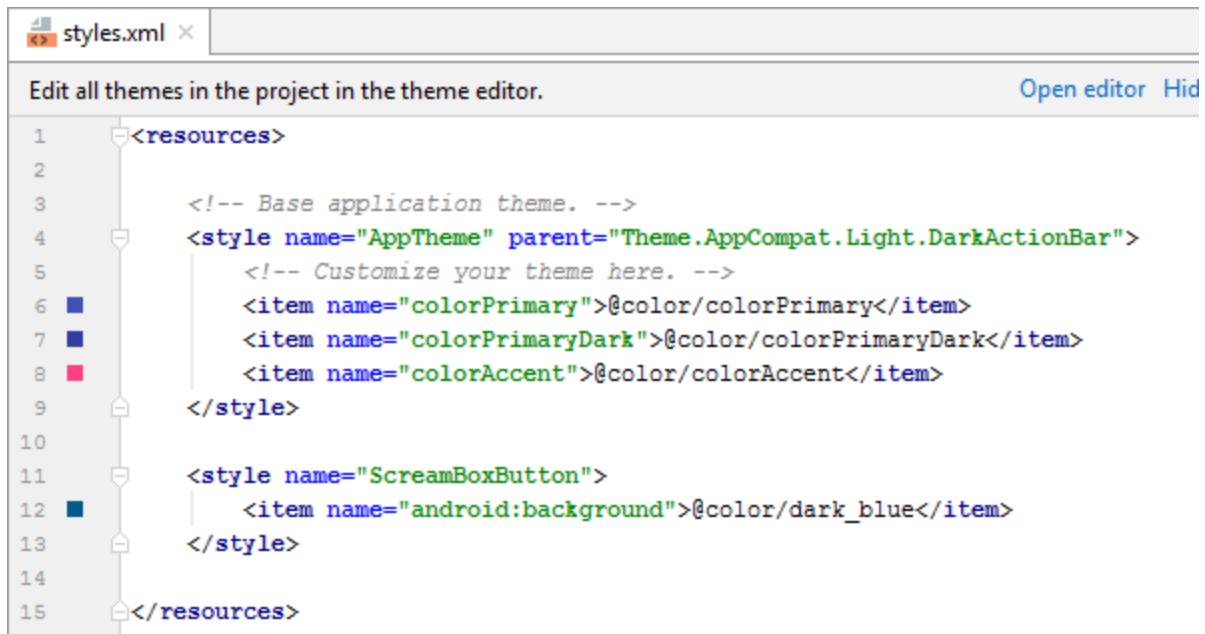
4. Color resources are a convenient way to specify color values in one place that you reference throughout your application.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="colorPrimary">#3F5185</color>
4     <color name="colorPrimaryDark">#303F9F</color>
5     <color name="colorAccent">#FF4081</color>
6
7     <color name="red">#F44336</color>
8     <color name="dark_red">#C3352B</color>
9     <color name="gray">#607D8B</color>
10    <color name="soothing_blue">#0083BF</color>
11    <color name="dark_blue">#005A8A</color>
12
13    <color name="purple_200">#FFBB86FC</color>
14    <color name="purple_500">#FF6200EE</color>
15    <color name="purple_700">#FF3700B3</color>
16    <color name="teal_200">#FF03DAC5</color>
17    <color name="teal_700">#FF018786</color>
18    <color name="black">#FF000000</color>
19    <color name="white">#FFFFFFFF</color>
20 </resources>
```

Styles

5. Now, update the buttons in ScreamBox with a *style*. A style is a set of attributes that you can apply to a widget.
6. Navigate to res/values/styles.xml and add a style named **ScreamBoxButton**.
7. When you created ScreamBox, your new project should have come with a built-in styles.xml file. If your project did not, create the file.



```
1 <resources>
2
3     <!-- Base application theme. -->
4     <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
5         <!-- Customize your theme here. -->
6         <item name="colorPrimary">@color/colorPrimary</item>
7         <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8         <item name="colorAccent">@color/colorAccent</item>
9     </style>
10
11     <style name="ScreamBoxButton">
12         <item name="android:background">@color/dark_blue</item>
13     </style>
14
15 </resources>
```

8. Here, you create a style called **ScreamBoxButton**. This style defines a single attribute, `android:background`, and sets it to a dark blue color. You can apply this style to as many widgets as you like and then update the attributes of all of those widgets in this one place.

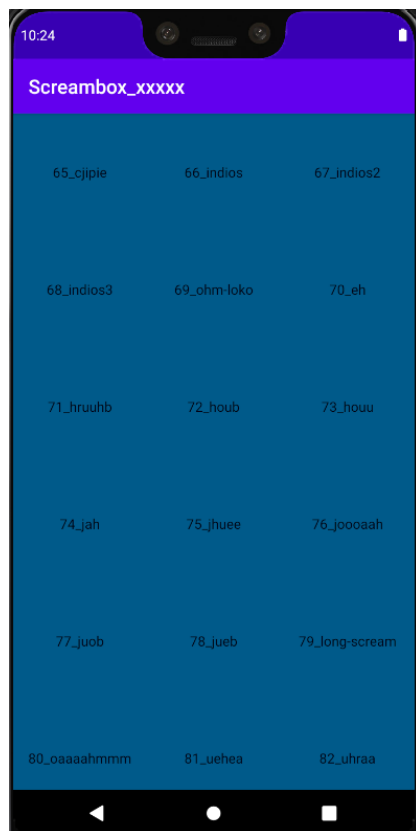


```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="id.ac.astra.polman.nimxxxxx.screambox">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="ScreamBox"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme">
12         <activity android:name=".MainActivity">
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />
17             </intent-filter>
18         </activity>
19     </application>
20
21 </manifest>
```

9. Now that the style is defined, apply **ScreamBoxButton** to your buttons in ScreamBox.

```
list_item_sound.xml x
10 <Button
11     style="@style/ScreamBoxButton"
12     android:layout_width="match_parent"
13     android:layout_height="120dp"
14     android:onClick="@{() -> viewModel.onButtonClicked()}"
15     android:text="@{viewModel.title}"
16     tools:text="Sound name"/>
17
18 </layout>
```

10. Run ScreamBox and you will see that all of your buttons now have a dark blue background color



Style inheritance

11. Styles also support inheritance: A style can inherit and override attributes from another style.
12. Create a new style called **ScreamBoxButton.Strong** that inherits from **ScreamBoxButton** and also bolds the text.

```

11 <style name="ScreamBoxButton">
12 |   <item name="android:background">@color/dark_blue</item>
13 </style>
14
15 <style name="ScreamBoxButton.Strong">
16 |   <item name="android:textStyle">bold</item>
17 </style>

```

13. While you could have added the `android:textStyle` attribute to the **ScreamBoxButton** style directly, you created **ScreamBoxButton.Strong** to demonstrate style inheritance.
14. The naming convention here is a little strange. When you name your style **ScreamBoxButton.Strong**, you are saying that your theme inherits attributes from **ScreamBoxButton**.
15. There is also an alternative inheritance naming style. You can specify a parent when declaring the style:

```

11 <style name="ScreamBoxButton">
12 |   <item name="android:background">@color/dark_blue</item>
13 </style>
14
15 <style name="StrongScreamBoxButton" parent="@style/ScreamBoxButton">
16 |   <item name="android:textStyle">bold</item>
17 </style>

```

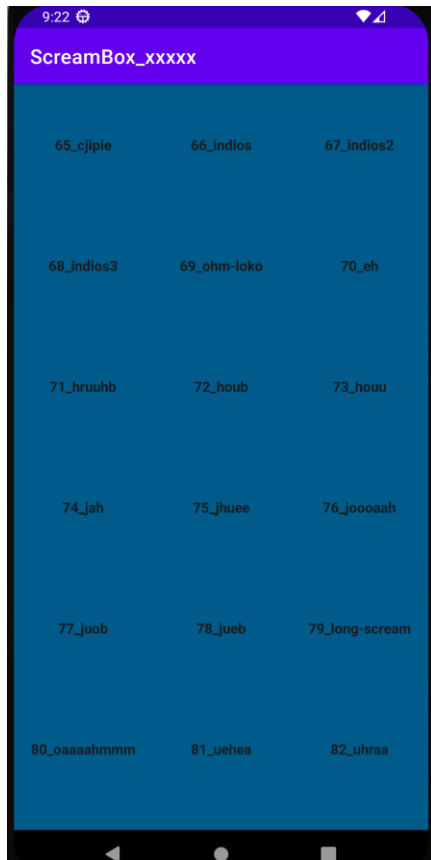
16. Stick with the **ScreamBoxButton.Strong** style in ScreamBox.
17. Update `list_item_sound.xml` to use your newer, stronger style.

```

10 <Button
11 |   style="@style/ScreamBoxButton.Strong"
12 |   android:layout_width="match_parent"
13 |   android:layout_height="120dp"
14 |   android:onClick="@{() -> viewModel.onButtonClicked()}"
15 |   android:text="@{viewModel.title}"
16 |   tools:text="Sound name"/>
17 </Button>
18 </layout>

```

18. Run ScreamBox and verify that your button text is indeed bold.



Themes

19. Styles are cool. They allow you to define a set of attributes in one place and then apply them to as many widgets as you want. The downside of styles is that you have to apply them to each and every widget, one at a time. What if you had a more complex app with lots of buttons in lots of layouts?
20. Adding your ScreamBoxButton style to them all could be a huge task.
21. That is where themes come in. Themes take styles a step further: They allow you to define a set of attributes in one place, like a style – but then those attributes are automatically applied throughout your app. Theme attributes can store a reference to concrete resources, such as colors, and they can also store a reference to styles.
22. In a theme, you can say, for example, “I want all buttons to use this style.” And then you do not need to find every button widget and tell it to use the theme.

Modifying the theme

23. When you created ScreamBox, it was given a default theme. Navigate to AndroidManifest.xml and look at the theme attribute on the application tag.

```
AndroidManifest.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="id.ac.astra.polman.screambox_XXXXXXXXXX">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="ScreamBox_XXXXXXXXXX"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportRtl="true"
11        android:theme="@style/AppTheme">
12    <activity android:name=".ScreamBoxActivity">
```

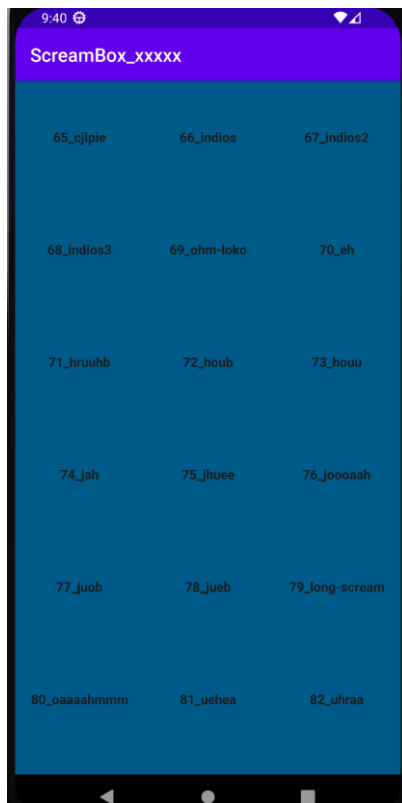
24. The theme attribute is pointing to a theme called **AppTheme**. **AppTheme** was declared in the styles.xml file that you modified earlier.
25. As you can see, a theme is also a style. But themes specify different attributes than a style does (as you will see in a moment). Themes are also given superpowers by being declared in the manifest. This is what causes the theme to be applied across the entire app automatically.
26. Navigate to the definition of the **AppTheme** theme by (Ctrl-clicking) on @style/AppTheme. Android Studio will take you to res/values/styles.xml.

```
styles.xml x
1 <resources>
2
3     <!-- Base application theme. -->
4     <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
5         <!-- Customize your theme here. -->
6         <item name="colorPrimary">@color/colorPrimary</item>
```

27. **AppTheme** is inheriting attributes from **Theme.AppCompat.Light.DarkActionBar**. Within **AppTheme**, you can add or override additional values from the parent theme.
28. The AppCompat library comes with three main themes:
 - a. **Theme.AppCompat** – a dark theme
 - b. **Theme.AppCompat.Light** – a light theme
 - c. **Theme.AppCompat.Light.DarkActionBar** – a light theme with a dark toolbar
29. Change the parent theme to **Theme.AppCompat** to give ScreamBox a dark theme as its base.

```
styles.xml x
1 <resources>
2
3     <!-- Base application theme. -->
4     <style name="AppTheme" parent="Theme.AppCompat">
5         <!-- Customize your theme here. -->
```

30. Run ScreamBox to see your new dark theme.



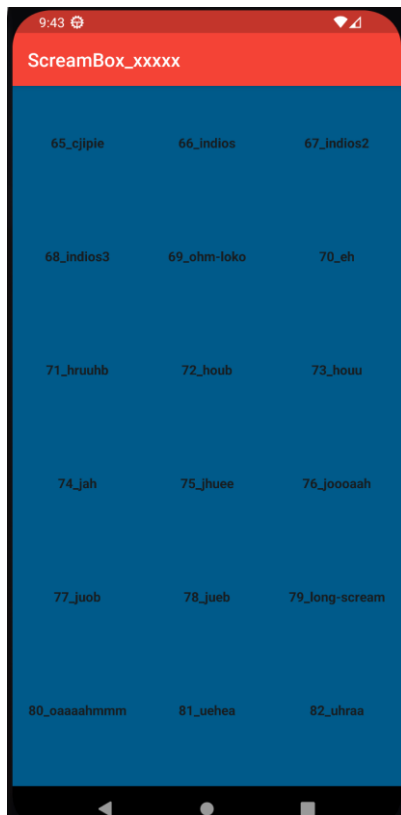
Adding Theme Colors

31. With the base theme squared away, it is time to customize the attributes of ScreamBox's **AppTheme**.
32. In the styles.xml file, you will see three attributes. Update them :



33. These theme attributes look similar to the style attributes that you set up earlier, but they specify different properties. Style attributes specify properties for an individual widget, such as the textStyle that you used to bold the button text. Theme attributes have a larger scope: They are properties that are set on the theme that any widget can access. For example, the toolbar will look at the colorPrimary attribute on the theme to set its background color.

34. These three attributes have a large impact. The `colorPrimary` attribute is the primary color for your app's brand. This color will be used as the toolbar's background and in a few other places.
35. `colorPrimaryDark` is used to color the status bar, which shows up at the top of the screen. Typically `colorPrimaryDark` will be a slightly darker version of your `colorPrimary` color. Status bar theming is a feature that was added to Android in Lollipop. Keep in mind that the status bar will be black on older devices (no matter what the theme specifies).
36. Finally, you set `colorAccent` to a gray color. `colorAccent` should contrast with your `colorPrimary` attribute; it is used to tint some widgets, such as an **EditText**.
37. You will not see the `colorAccent` attribute affect `ScreamBox` because **Buttons** do not support tinting. You still specify `colorAccent` because it is a good idea to think about these three color attributes together. These colors should mesh, and the default `colorAccent` attribute from your parent theme may clash with the other colors that you specified. This sets you up well for any future additions.
38. Run `ScreamBox` to see the new colors in action. Your app should look like this:



Overriding Theme Attributes

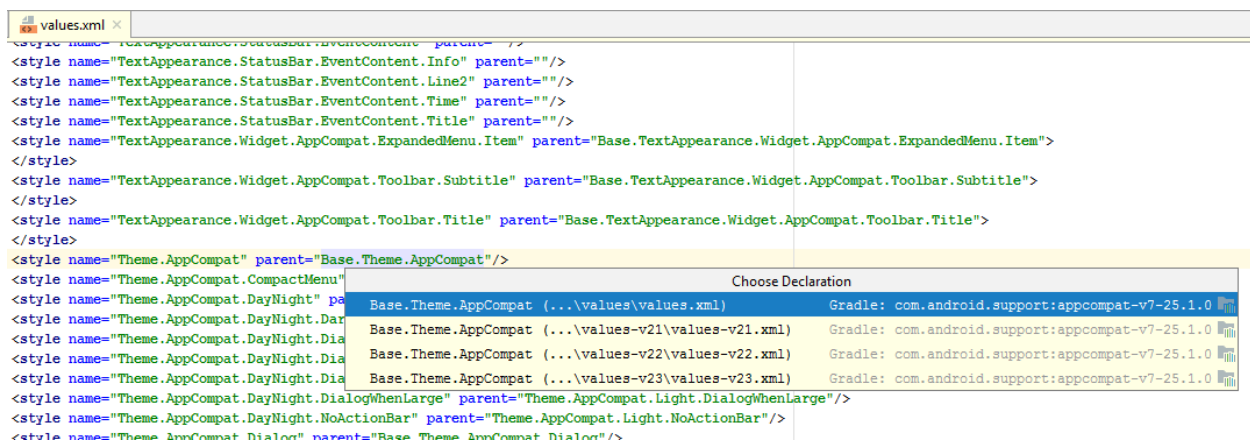
39. Now that the colors are worked out, it is time to dive in and see what theme attributes exist that you can override. Be warned, theme spelunking is tough. There is little to no documentation about which attributes exist, which ones you can override yourself, and even what the attributes do.
40. Your first goal is to change the background color of `ScreamBox` by altering the theme. While you could navigate to `res/layout/fragment_scream_box.xml` and manually set the `android:background` attribute on your **RecyclerView** – and then repeat the process in every other fragment and activity

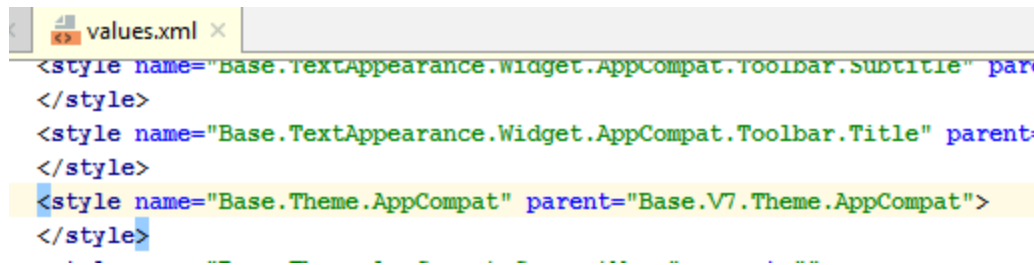
layout file that might exist – this would be wasteful. Wasteful of your time, obviously, but also wasteful of app effort.

41. The theme is always setting a background color. By setting another color on top of that, you are doing extra work. You are also writing code that is hard to maintain by duplicating the background attribute throughout the app.

Theme spelunking

42. Instead, you want to override the background color attribute on your theme. To discover the name of this attribute, take a look at how this attribute is set by your parent theme: **Theme.AppCompat**.
43. You might be thinking, “How will I know which attribute to override if I don’t know its name?” You will not. You will read the names of the attributes and you will think, “That sounds right.” Then you will override that attribute, run the app, and hope that you chose wisely.
44. What you want to do is search through the ancestors of your theme. To do this, you will keep on navigating up to one parent after another until you find a suitable attribute. Open your styles.xml file and (**Ctrl-click**) on **Theme.AppCompat**. Let’s see how deep the rabbit hole goes.
45. If you are unable to navigate through your theme attributes directly in Android Studio, or you want to do this outside of Android Studio, you can find Android’s theme sources in the directory your-SDKdirectory/platforms/android-24/data/res/values directory.
46. At the time, you are brought to a very large file with a focus on this line:
`<style name="Theme.AppCompat" parent="Base.Theme.AppCompat" />`
47. The theme, **Theme.AppCompat**, inherits attributes from **Base.Theme.AppCompat**. Interestingly, **Theme.AppCompat** does not override any attributes itself. It just points to its parent.
48. (**Ctrl-click**) on **Base.Theme.AppCompat**. Android Studio will tell you that this theme is resource qualified. There are a few different versions of this theme depending on the version of Android that you are on.
49. Choose the values/values.xml version and you will be brought to **Base.Theme.AppCompat**’s definition.





```

<style name="Base.TextAppearance.Widget.AppCompat.Toolbar.Subtitle" parent="Base.TextAppearance.Widget.AppCompat.Toolbar.Subtitle" />
</style>
<style name="Base.TextAppearance.Widget.AppCompat.Toolbar.Title" parent="Base.TextAppearance.Widget.AppCompat.Toolbar.Title" />
</style>
<style name="Base.Theme.AppCompat" parent="Base.V7.Theme.AppCompat">
</style>

```

50. **Base.Theme.AppCompat** is another theme that exists only for its name and does not override any attributes. Continue along to its parent theme: **Base.V7.Theme.AppCompat**.



```

<style name="Base.V7.Theme.AppCompat" parent="Platform.AppCompat">
    <item name="windowNoTitle">false</item>
    <item name="windowActionBar">true</item>
    <item name="windowActionBarOverlay">false</item>
    <item name="windowActionModeOverlay">false</item>
    <item name="actionBarPopupTheme">@null</item>
</style>

```

51. You are getting closer. Scan through the list of attributes in **Base.V7.Theme.AppCompat**.

52. You will not see an attribute that seems to change the background color. Navigate to **Platform.AppCompat**. You will see that this is resource qualified. Again, choose the values/values.xml version.



```

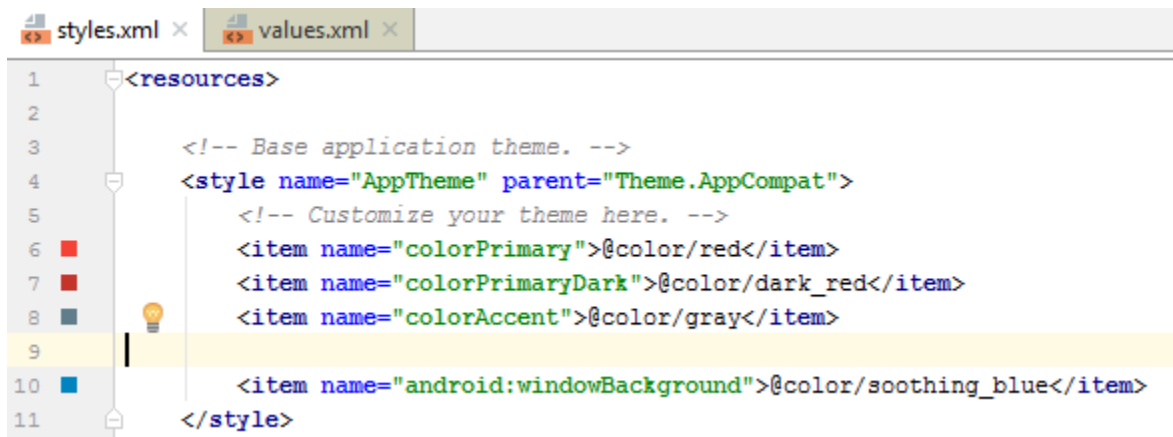
1280 <style name="Platform.AppCompat" parent="android:Theme.Holo">
1281     <item name="android:windowNoTitle">true</item>
1282     <item name="android:windowActionBar">false</item>
1283
1284     <item name="android:buttonBarStyle">?attr/buttonBarStyle</item>
1285     <item name="android:buttonBarButtonStyle">?attr/buttonBarButtonStyle</item>
1286     <item name="android:borderlessButtonStyle">?attr/borderlessButtonStyle</item>
1287
1288     <!-- Window colors -->
1289     <item name="android:colorForeground">@color/foreground_material_dark</item>
1290     <item name="android:colorForegroundInverse">@color/foreground_material_light</item>
1291     <item name="android:colorBackground">@color/background_material_dark</item>
1292     <item name="android:colorBackgroundCacheHint">@color/abc_background_cache_hint_sel</item>
1293     <item name="android:disabledAlpha">@dimen/abc_disabled_alpha_material_dark</item>
1294     <item name="android:backgroundDimAmount">0.6</item>
1295     <item name="android:windowBackground">@color/background_material_dark</item>

```

53. Here you see that the parent of the **Platform.AppCompat** theme is **android:Theme.Holo**. Notice that the parent theme is not referenced just as **Theme**. Instead it has the **android** namespace in front of it.

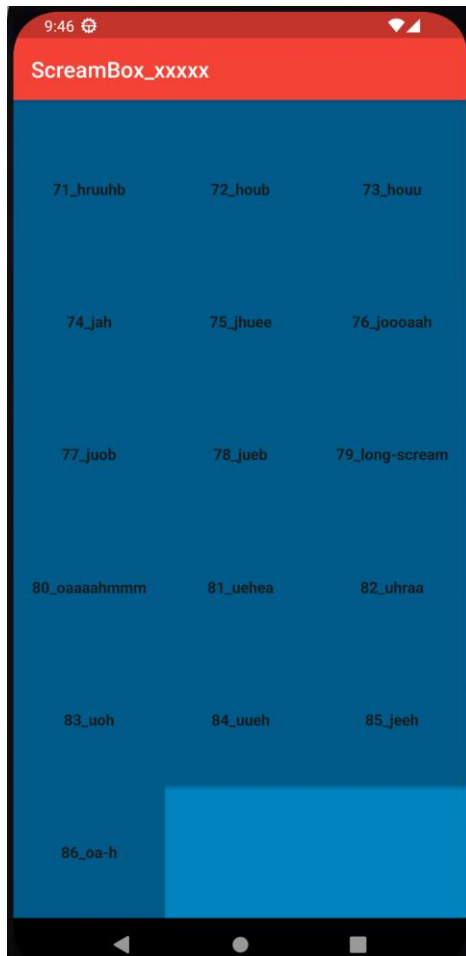
54. You can think of the **AppCompat** library as something that lives within your own app. When you build your project, you include the **AppCompat** library and it brings along a bunch of Java and XML files. Those files are just like the files that you wrote yourself. If you want to refer to something in the **AppCompat** library, you do it directly. You would just write **Theme.AppCompat**, because those files exist in your app.

55. Themes that exist in the Android OS, like **Theme**, have to be declared with the namespace that points to their location. The AppCompatActivity library uses **android:Theme** because the theme exists in the Android OS.
56. You have finally arrived. Here you see many more attributes that you can override in your theme. You can of course navigate to **Platform.AppCompat**'s parent, **Theme**, but this is not necessary.
57. You will find the attribute you need in this theme.
58. Right near the top, `windowBackground` is declared. It seems likely that this attribute is the background for the theme.
59. This is the attribute that you want to override in `ScreamBox`. Navigate back to your `styles.xml` file and override the `windowBackground` attribute.



```
1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat">
5 <!-- Customize your theme here. -->
6 <item name="colorPrimary">@color/red</item>
7 <item name="colorPrimaryDark">@color/dark_red</item>
8 <item name="colorAccent">@color/gray</item>
9
10 <item name="android:windowBackground">@color/soothing_blue</item>
11 </style>
```

60. Notice that you must use the `android` namespace when overriding this attribute, because `windowBackground` is declared in the Android OS.
61. Run `ScreamBox`, scroll down to the bottom of your `recycler view`, and verify that the background (where it is not covered with a button) is a soothing blue.



62. The steps that you just went through to find the `windowBackground` attribute are the same steps that every Android developer takes when modifying an app's theme. You will not find much documentation on these attributes. Most people go straight to the source to see what is available.
63. To recap, you navigated through the following themes:
- a. **Theme.AppCompat**
 - b. **Base.Theme.AppCompat**
 - c. **Base.V7.Theme.AppCompat**
 - d. **Platform.AppCompat**
64. You navigated through the theme hierarchy until you arrived at `AppCompat`'s root theme. As you become more familiar with your theme options, you may opt to skip ahead to the appropriate theme in the future. But it is nice to follow the hierarchy so you can see your theme's roots.
65. Be aware that this theme hierarchy may change over time. But the task of walking the hierarchy will not. You follow your theme hierarchy until you find the attribute that you want to override.

Modifying Button Attributes

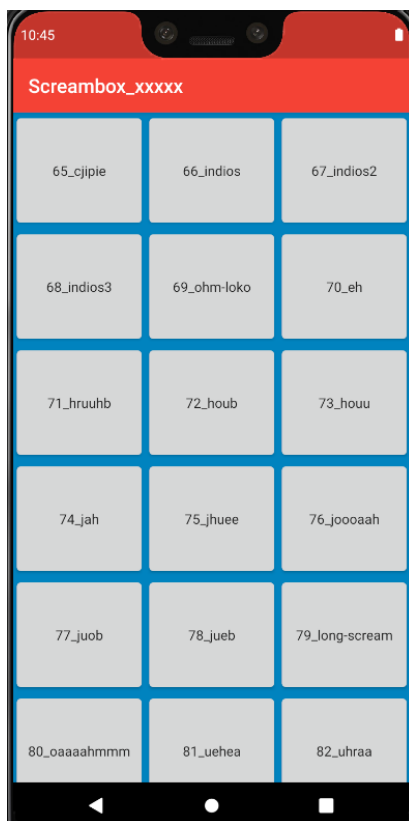
66. Earlier you customized the buttons in `ScreamBox` by manually setting a style attribute in the `res/layout/list_item_sound.xml` file. If you have a more complex app, with buttons throughout many

fragments, setting a style attribute on each and every button does not scale well. You can take your theme a step further by defining a style in your theme for every button in your app.

67. Before adding a button style to your theme, remove the style attribute from your res/layout/list_item_sound.xml file.

```
list_item_sound.xml x
10 <Button
11     android:layout_width="match_parent"
12     android:layout_height="120dp"
13     android:onClick="@{() -> viewModel.onButtonClicked()}"
14     android:text="@{viewModel.title}"
15     tools:text="Sound name"/>
16
17 </layout>
```

68. Run ScreamBox again and verify that your buttons are back to the old, bland look.



69. Go theme spelunking again. This time, you are looking for buttonStyle. You will find it in **Base.V7.Theme.AppCompat**.

```
values.xml x
655 <style name="Base.V7.Theme.AppCompat" parent="Platform.AppCompat">
656     <item name="windowNoTitle">false</item>
657     <item name="windowActionBar">true</item>
```

....

```

784      <!-- Button styles -->
785      <item name="buttonStyle">@style/Widget.AppCompat.Button</item>
786      <item name="buttonStyleSmall">@style/Widget.AppCompat.Button.Small</item>
787      <item name="android:textAppearanceButton">@style/TextAppearance.AppCompat.Widget.Button</item>

```

70. **buttonStyle** specifies the style of any normal button within your app. The **buttonStyle** attribute points to a style resource rather than a value. When you updated the **windowBackground** attribute, you passed in a value: the color. In this case, **buttonStyle** should point to a style. Navigate to **Widget.AppCompat.Button** to see the button style.

```
<style name="Widget.AppCompat.Button" parent="Base.Widget.AppCompat.Button"/>
```

71. **Widget.AppCompat.Button** does not define any attributes itself. Navigate to its parent to see the goods. You will find that there are two versions of the base style. Choose the values/values.xml version.

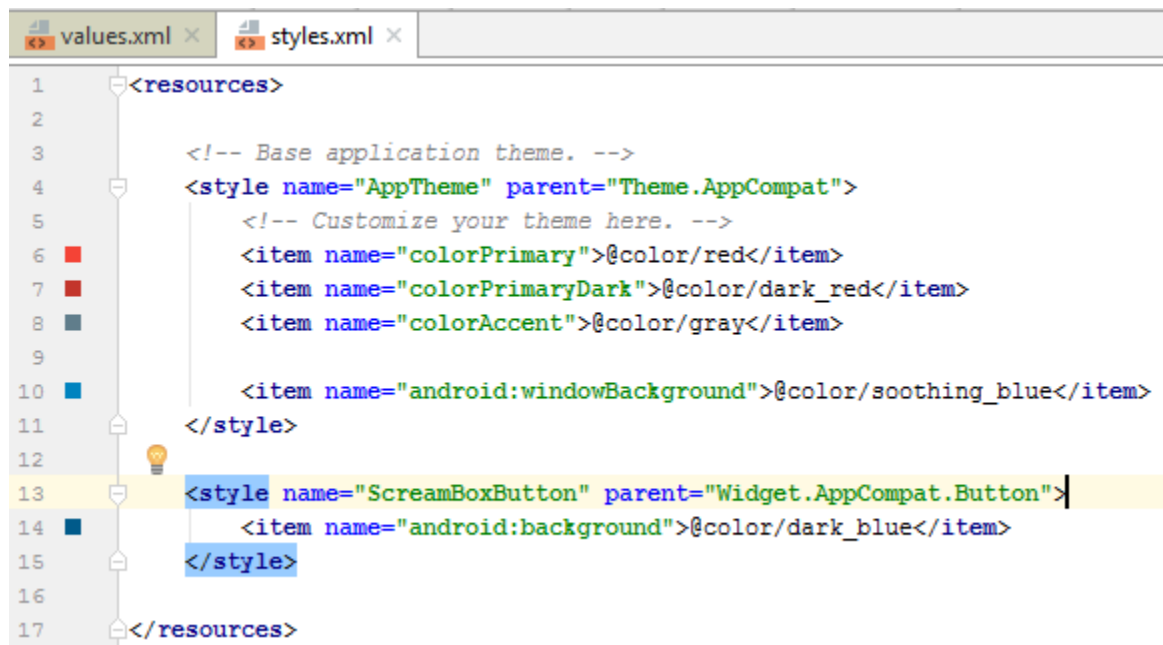
```

<style name="Base.Widget.AppCompat.Button" parent="android:Widget">
    <item name="android:background">@drawable/abc_btn_default_mtrl_shape</item>
    <item name="android:textAppearance">?android:attr/textAppearanceButton</item>
    <item name="android:minHeight">48dip</item>
    <item name="android:minWidth">88dip</item>
    <item name="android:focusable">true</item>
    <item name="android:clickable">true</item>
    <item name="android:gravity">center_vertical|center_horizontal</item>
</style>

```

72. Every **Button** that you use in **ScreamBox** is given these attributes.

73. Duplicate what happens in Android's own theme in **ScreamBox**. Change the parent of **ScreamBoxButton** to inherit from the existing button style. Also, remove your **ScreamBoxButton.Strong** style from earlier.

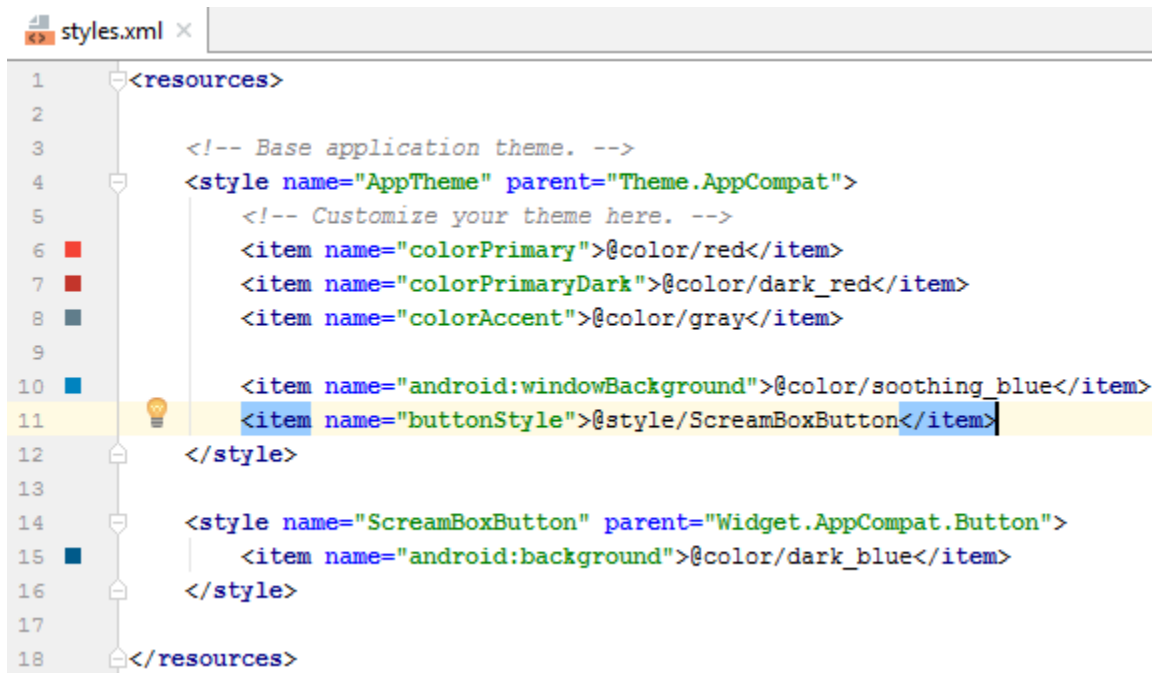


```

1  <resources>
2
3      <!-- Base application theme. -->
4      <style name="AppTheme" parent="Theme.AppCompat">
5          <!-- Customize your theme here. -->
6          <item name="colorPrimary">@color/red</item>
7          <item name="colorPrimaryDark">@color/dark_red</item>
8          <item name="colorAccent">@color/gray</item>
9
10         <item name="android:windowBackground">@color/soothing_blue</item>
11     </style>
12
13     <style name="ScreamBoxButton" parent="Widget.AppCompat.Button">
14         <item name="android:background">@color/dark_blue</item>
15     </style>
16
17 </resources>

```

74. You specified a parent of **Widget.AppCompat.Button**. You want your button to inherit all of the properties that a normal button would receive and then selectively modify attributes.
75. If you do not specify a parent theme for **ScreamBoxButton**, you will notice that your buttons devolve into something that does not look like a button at all. Properties you expect to see, such as the text centered in the button, will be lost.
76. Now that you have fully defined **ScreamBoxButton**, it is time to use it. Look back at the `buttonStyle` attribute that you found earlier when digging through Android's themes. Duplicate this attribute in your own theme.



```
1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat">
5     <!-- Customize your theme here. -->
6     <item name="colorPrimary">@color/red</item>
7     <item name="colorPrimaryDark">@color/dark_red</item>
8     <item name="colorAccent">@color/gray</item>
9
10    <item name="android:windowBackground">@color/soothing_blue</item>
11    <item name="buttonStyle">@style/ScreamBoxButton</item>
12 </style>
13
14 <style name="ScreamBoxButton" parent="Widget.AppCompat.Button">
15     <item name="android:background">@color/dark_blue</item>
16 </style>
17
18 </resources>
```

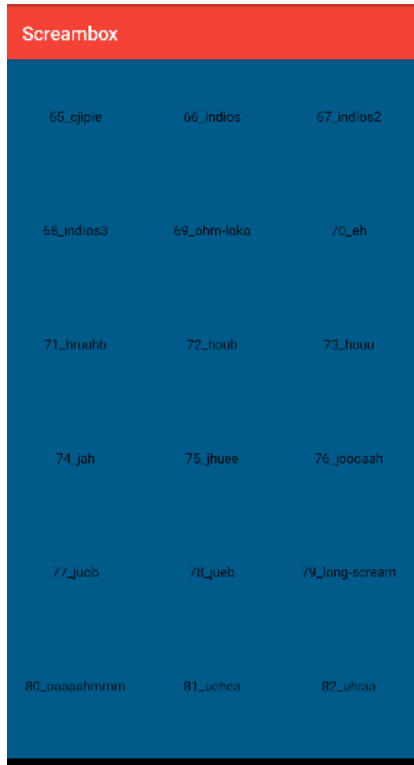
77. Notice that you do not use the `android:` prefix when defining `buttonStyle`. This is because the `buttonStyle` attribute that you are overriding is implemented in the `AppCompat` library.



```
11 <Button
12     style="@style/ScreamBoxButton"
13     android:layout_width="match_parent"
14     android:layout_height="120dp"
15     android:onClick="@{() -> viewModel.onButtonClicked()}"
16     android:text="@{viewModel.title}"
17     tools:text="Sound name"
18 />
```

78. You are now overriding the `buttonStyle` attribute and substituting your own style: **ScreamBoxButton**.

79. Run ScreamBox and notice that all of your buttons are dark blue. You changed the look of every normal button in ScreamBox without modifying any layout files directly.



Notes:

1. Create folder PRG6_M9_P3_XXXXXXXXXXXX.
2. Zip the folder and submit it to the server.

Bibliography:

- Marsicano, et. al., “Android Programming – The Big Nerd Ranch”, 5th Ed, 2022, Pearson Technology.