Styles and Themes

https://developer.android.com/guide/topics/ui/look-and-feel/themes

Cont ...

- Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design.
- A **style** is a collection of attributes that specify the appearance for a single View.
- A style can specify attributes such as font color, font size, background color, and much more.

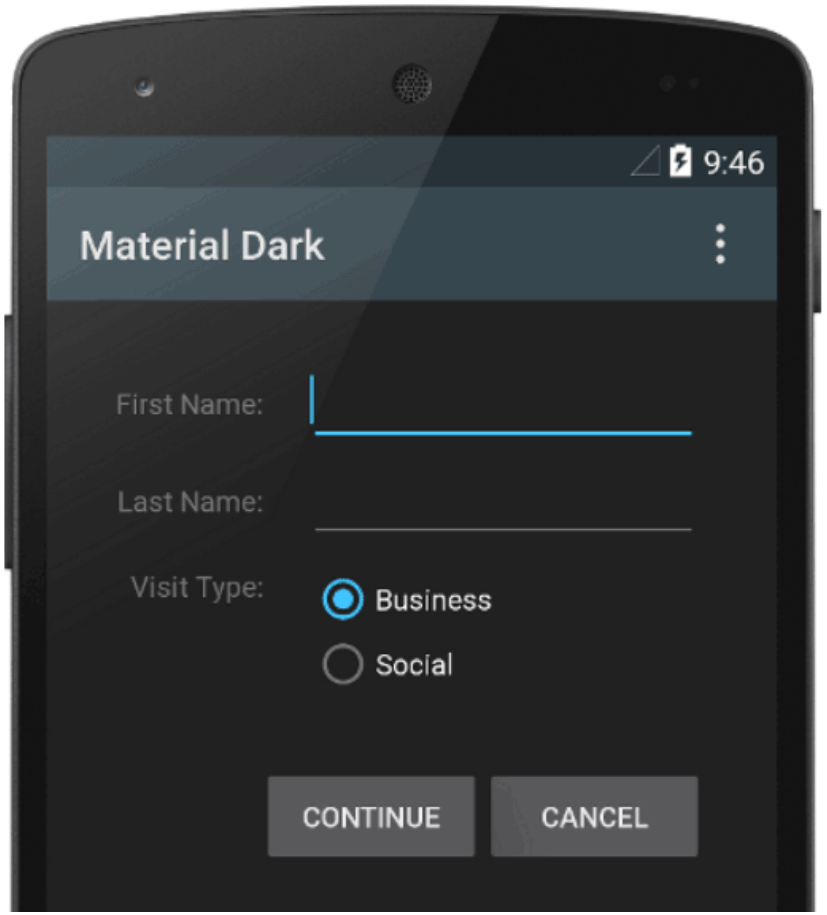
Cont ...

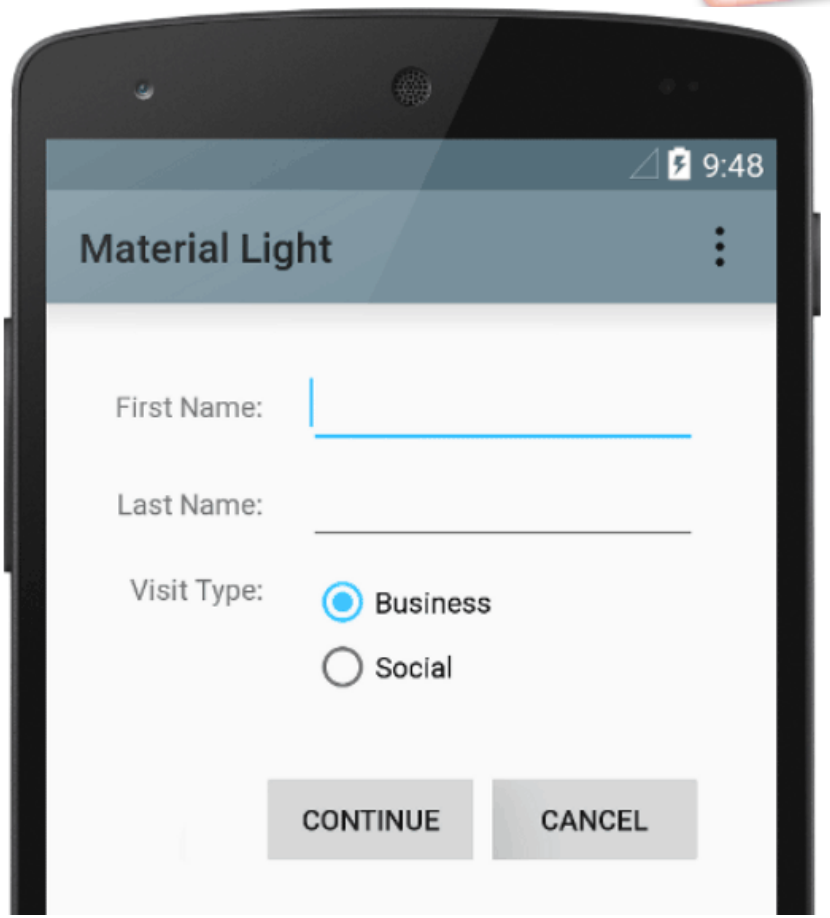
- A **theme** is a type of style that's applied to an entire app, activity, or view hierarchy—not just an individual view.
- When you apply your style as a theme, every view in the app or activity applies each style attribute that it supports.
- Themes can also apply styles to non-view elements, such as the status bar and window background.

Cont ...

- Styles and themes are declared in a style resource file in res/values/, usually named styles.xml.

Cont ...


9:46
Material Dark
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL


9:48
Material Light
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL

Create and apply a style

- To create a new style or theme, open your project's res/values/styles.xml file.
- For each style you want to create, follow these steps:
  - Add a <style> element with a name that uniquely identifies the style.
  - Add an <item> element for each style attribute you want to define. The name in each item specifies an attribute you would otherwise use as an XML attribute in your layout. The value in the <item> element is the value for that attribute.

Cont ...

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

Cont ...

- You can apply the style to a view as follows:

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

- However, instead of applying a style to individual views, you'll usually apply styles as a theme for your entire app, activity, or collection of views.

Extend and customize a style

- When creating your own styles, you should always extend an existing style from the framework or support library so that you maintain compatibility with platform UI styles.
- To extend a style, specify the style you want to extend with the parent attribute.
- You can then override the inherited style attributes and add new ones.

Cont ...

- For example, you can inherit the Android platform's default text appearance and modify it as follows:

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

Cont ...

- However, you should always inherit your core app styles from the Android Support Library.
- The styles in the support library provide compatibility with Android 4.0 (API level 14) and higher by optimizing each style for the UI attributes available in each version.
- The support library styles often have a name similar to the style from the platform, but with AppCompat included.

Apply a style as a theme

- You can create a theme the same way you create styles.
- The difference is how you apply it: instead of applying a style with the style attribute on a view, you apply a theme with the android:theme attribute on either the <application> tag or an <activity> tag in the AndroidManifest.xml file.

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

Cont ...

- And here's how to apply the "light" theme to just one activity:

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

Cont ...

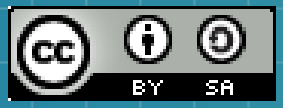
- Now every view in the app or activity applies the styles defined in the given theme.
- If a view supports only some of the attributes declared in the style, then it applies only those attributes and ignores the ones it does not support.
- Beginning with Android 5.0 (API level 21) and Android Support Library v22.1, you can also specify the android:theme attribute to a view in your layout file.
- This modifies the theme for that view and any child views, which is useful for altering theme color palettes in a specific portion of your interface.

Style hierarchy

- Android provides a variety of ways to set attributes throughout your Android app.
- When choosing how to style your app, be mindful of Android's style hierarchy.
- In general, you should use themes and styles as much as possible for consistency.
- If you've specified the same attributes in multiple places, the list below determines which attributes are ultimately applied.

Cont ...

- The list is ordered from highest precedence to lowest:
  - Applying character- or paragraph-level styling via text spans to TextView-derived classes
  - Applying attributes programmatically
  - Applying individual attributes directly to a View
  - Applying a style to a View
  - Default styling
  - Applying a theme to a collection of Views, an activity, or your entire app
  - Applying certain View-specific styling, such as setting a TextAppearance on a TextView



Styles and Themes

https://developer.android.com/guide/topics/ui/look-and-feel/themes

Cont ...

- Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design.
- A **style** is a collection of attributes that specify the appearance for a single View.
- A style can specify attributes such as font color, font size, background color, and much more.

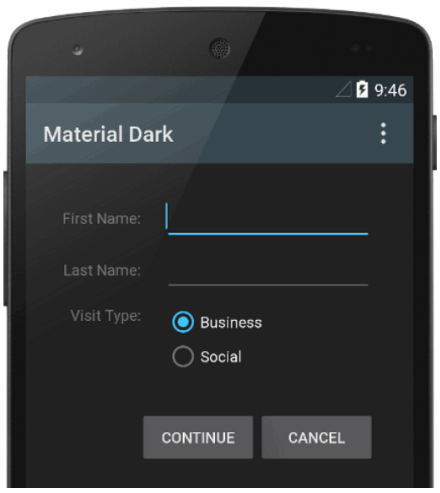
Cont ...

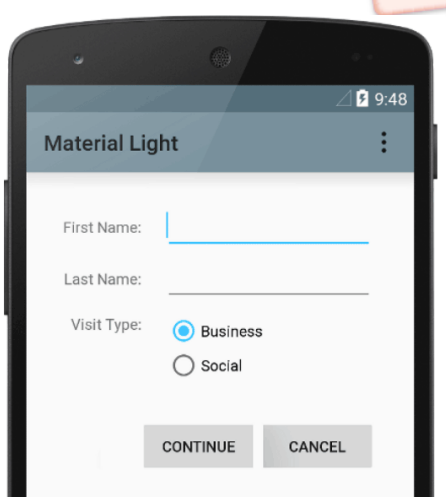
- A **theme** is a type of style that's applied to an entire app, activity, or view hierarchy—not just an individual view.
- When you apply your style as a theme, every view in the app or activity applies each style attribute that it supports.
- Themes can also apply styles to non-view elements, such as the status bar and window background.

Cont ...

- Styles and themes are declared in a style resource file in res/values/, usually named styles.xml.

Cont ...


9:46
Material Dark
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL


9:48
Material Light
First Name:
Last Name:
Visit Type:
Business
Social
CONTINUE
CANCEL

Create and apply a style

- To create a new style or theme, open your project's res/values/styles.xml file.
- For each style you want to create, follow these steps:
  - Add a <style> element with a name that uniquely identifies the style.
  - Add an <item> element for each style attribute you want to define. The name in each item specifies an attribute you would otherwise use as an XML attribute in your layout. The value in the <item> element is the value for that attribute.

Cont ...

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

Cont ...

- You can apply the style to a view as follows:

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

- However, instead of applying a style to individual views, you'll usually apply styles as a theme for your entire app, activity, or collection of views.

Extend and customize a style

- When creating your own styles, you should always extend an existing style from the framework or support library so that you maintain compatibility with platform UI styles.
- To extend a style, specify the style you want to extend with the parent attribute.
- You can then override the inherited style attributes and add new ones.

Cont ...

- For example, you can inherit the Android platform's default text appearance and modify it as follows:

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

Cont ...

- However, you should always inherit your core app styles from the Android Support Library.
- The styles in the support library provide compatibility with Android 4.0 (API level 14) and higher by optimizing each style for the UI attributes available in each version.
- The support library styles often have a name similar to the style from the platform, but with AppCompat included.

Apply a style as a theme

- You can create a theme the same way you create styles.
- The difference is how you apply it: instead of applying a style with the style attribute on a view, you apply a theme with the android:theme attribute on either the <application> tag or an <activity> tag in the AndroidManifest.xml file.

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

Cont ...

- And here's how to apply the "light" theme to just one activity:

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

Cont ...

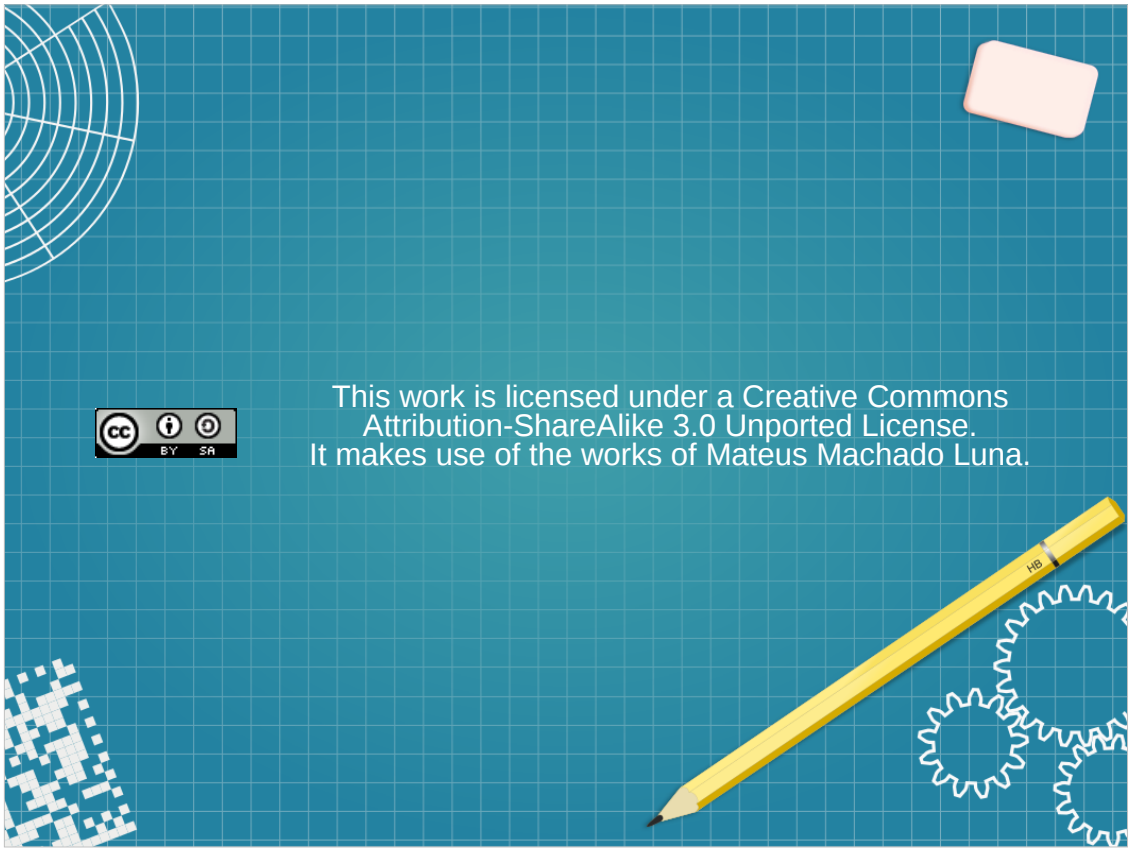
- Now every view in the app or activity applies the styles defined in the given theme.
- If a view supports only some of the attributes declared in the style, then it applies only those attributes and ignores the ones it does not support.
- Beginning with Android 5.0 (API level 21) and Android Support Library v22.1, you can also specify the android:theme attribute to a view in your layout file.
- This modifies the theme for that view and any child views, which is useful for altering theme color palettes in a specific portion of your interface.

Style hierarchy

- Android provides a variety of ways to set attributes throughout your Android app.
- When choosing how to style your app, be mindful of Android's style hierarchy.
- In general, you should use themes and styles as much as possible for consistency.
- If you've specified the same attributes in multiple places, the list below determines which attributes are ultimately applied.

Cont ...

- The list is ordered from highest precedence to lowest:
  - Applying character- or paragraph-level styling via text spans to TextView-derived classes
  - Applying attributes programmatically
  - Applying individual attributes directly to a View
  - Applying a style to a View
  - Default styling
  - Applying a theme to a collection of Views, an activity, or your entire app
  - Applying certain View-specific styling, such as setting a TextAppearance on a TextView


This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Unported License.
It makes use of the works of Mateus Machado Luna.
BY SA
HB