Queens University Belfast

Android Studio

CSC3054 / CSC7054

Create a Style / Theme for your App

Styles

Android Style works in a very similar way to Cascading Style Sheets (CSS) that are used in web design. There are number of attributes with each Android view that you can set to change the look and feel for your application. A style can specify properties such as height, padding, font colour, font size, background colour, and much more.

Assigning a style to one View

If you want to assign attributes to one view only then the simple approach would be to define the style attributes for every attribute separately as shown in figure 1. This is not good for source code maintenance.

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"
    android:textColor="#1B7E5A"
    android:typeface="serif"
    android:textStyle="bold"/>
```

FIGURE 1 - ASSIGNING SEPARATE STYLE ATTRIBUTES

Defining Styles

A style is defined in an XML resource that is separate from the XML that specifies the layout. The XML file resides under the `res/values/` directory of your project and will have `<resource>` as the root node which is mandatory for the style file. The name of this file is `style.xml`. You can define multiple styles per file using `<style>` tag but each style will have a name that uniquely identifies that style. Android style attributes are set using `<item>` tag. The value for the `<item>` can be a keyword string, a hex colour, a reference to another type, or other value depending on the style property.

```
<resources xmlns:android="http://schemas.android.com/apk/res/android">

    <style name="LoginText">
        <item name="android:textColor">#1B7E5A</item>
        <item name="android:textStyle">bold</item>
        <item name="android:typeface">serif</item>
    </style>

</resources>
```

FIGURE 2 - STYLE.XML

Using Style

Once your style is defined, you can use it in your XML Layout file using `style` attribute as shown in figure 3.

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world"
    style="@style/LoginText"/>
```

FIGURE 3 - USING THE STYLE.XML

When the app is run, it should look like figure 4:

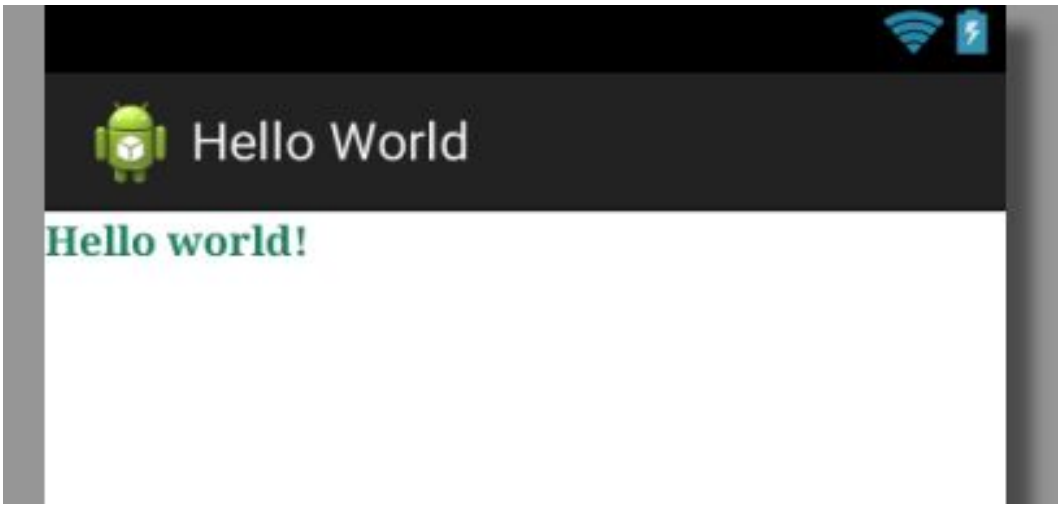

FIGURE 4 - RUNNING THE APP

Style Inheritance

Android supports style inheritance in very much similar way as CSS does in web design as shown in figure5.

```
<style name="CustomButtonStyle">
    <item name="android:background">#1B7E5A</item>
    <item name="android:textColor">#494949</item>
    <item name="android:textSize">10sp</item>
    <item name="android:textStyle">bold</item>
    <item name="android:gravity">center</item>
    <item name="android:shadowColor">#000000</item>
</style>


<style name="CustomButtonStyle.LargeFont">
    <item name="android:textSize">20sp</item>
</style>

<style name="CustomButtonStyle.LargeFont.Red">
    <item name="android:textColor">#ff0000</item>
</style>
```

FIGURE 5 STYLE INHERITANCE

You can inherit properties from an existing `style.xml` file in your `activity_main.xml` file. You can define only the properties you want to change or add as shown in figure 6.

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@+id/textView1"
    android:layout_below="@+id/textView1"
    android:layout_marginTop="21dp"
    android:text="Button1"
    style="@style/CustomButtonStyle"/>

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/button1"
    android:layout_centerHorizontal="true"
    android:text="Button2"
    style="@style/CustomButtonStyle.LargeFont"/>

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/button2"
    android:layout_alignBottom="@+id/button2"
    android:layout_alignParentRight="true"
    android:layout_marginRight="17dp"
    android:text="Button3"
    style="@style/CustomButtonStyle.LargeFont.Red"/>
```

FIGURE 6 APPLYING STYLE INHERITANCE

When you run the application your custom style is applied as shown in figure 7.

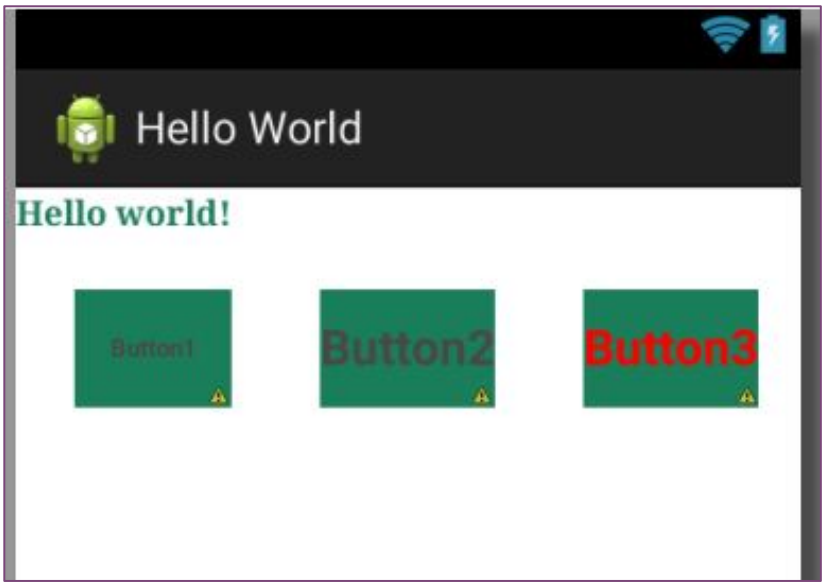

FIGURE 7 - APPLIED STYLE

Themes

A theme is nothing but an Android style applied to an entire `Activity` or application, rather than an individual `View`. Thus, 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 same `CustomFontStyle` as a theme for an activity and then all the text inside that `Activity` will have that type of font.

Setting a Theme

To set a theme for all the activities in your application, open the `AndroidManifest.xml` file and edit the `<application>` tag to include the `android:theme` attribute with the style name.

```
<application android:theme = "@style/CustomFontStyle" >
```

But if you want a theme applied to just one activity in your application, then add the `android:theme` attribute to the `<activity>` tag only.

```
<activity android:theme = "@style/CustomFontStyle" >
```