Style Definition

Introduction

- We can assign a separated style definition for each view object we are working with.
- The style definition is a collection of properties such as height, padding, font color, font size, background color and others. These properties specify the look and the format of a specific view.

Style Definition

- The style definition is defined in a separated XML document we should save within the `res\values` folder. The style definition isn't part of the layout XML document.

Simple Style Definition

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />

<TextView
android:id="@+id/TextView01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="hello earth"
style="@style/MyImportantFont" />

</LinearLayout>
```

The Style Definition Resource

Simple Style Definition

```
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int TextView01=0x7f060000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f050001;
        public static final int hello=0x7f050000;
    }
    public static final class style {
        public static final int MyImportantFont=0x7f040000;
    }
}
```


The Style Definition Resource

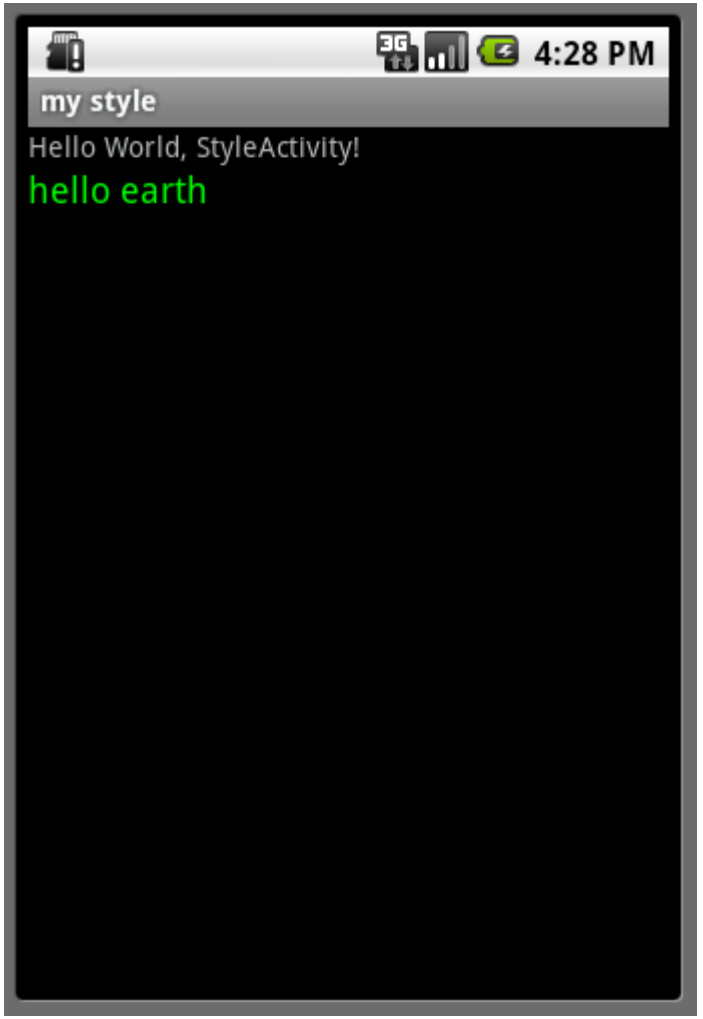
Simple Style Definition

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyImportantFont"
        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">sans</item>
    </style>
</resources>
```

The Style Definition Resource

The parent attribute in the <style> element is optional. It specifies the resource ID of another style from which this style should inherit properties. These inherited properties can be overridden

Simple Style Definition



Style Definition Inheritance

- When we define a style we can specify that it should inherit from another already defined style.
- We can then override the style definitions we want to change.

Style Definition Inheritance

- When dealing with inheritance from a style definition defined by the android platform we should use the parent attribute.

```
<style name="RedText" parent="@android:style/TextAppearance">
        <item name="android:textColor">#FF3300</item>
</style>
```

Style Definition Inheritance

- Instead of using the parent attribute for specifying the parent style (e.g. parent=”ParentStyle”) from which the new defined style (e.g. “TheNewStyle”) extends, we can name the new defined style in the following way:

```
<style name="ParentStyle.TheNewStyle">
        <item name="android:textColor">#00FF00</item>
</style>
```

- We will refer this new style by writing its full qualified name:

`@style/ParentStyle.TheNewStyle`.

Style Definition Inheritance

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="parentstyle">
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">sans</item>
        <item name="android:textSize">20px</item>
    </style>
</resources>
```

Style Definition Inheritance

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="parentstyle.jaja">
        <item name="android:textStyle">italic</item>
        <item name="android:textSize">40px</item>
    </style>
</resources>
```

Style Definition Inheritance

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <TextView
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello"
      style="@style/parentstyle.jaja"
    />
</LinearLayout>
```

Style Definition Inheritance

```
package com.absle;

import android.app.Activity;
import android.os.Bundle;

public class GogoActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Style Definition Inheritance

Style Definition Inheritance

- We can continue the inheritance to as many levels as we want by chaining style names separated with periods.

```
<style name="GrandParentStyle.ParentStyle.ChildStyle">
    <item name="android:textSize">8sp</item>
</style>
```

- We will refer this new style as

```
@style/GrandParentStyle.ParentStyle.ChildStyle
```

Style Properties

- You can find the properties that apply for each view browsing the corresponding class reference that lists all attributes supported by an XML document.
- When trying to apply a specific property the view doesn't support the view will simply ignore it.
- When applying a style definition to a ViewGroup the child view elements do not inherit the style definition properties.

Style Properties

Theme Definition

- A theme is a style definition for the entire activity or the application.
- When applying a style definition for the entire activity (or application) it will apply for each one of the views the activity (or application) includes.
- We define a theme the same way we define a simple style for a specific view.

Theme Definition

- Some of the style properties cannot apply for views. They can apply for the entire window only. Among these properties you can find ones that allow us to hide the application title, hide the status bar, or even change the window's background.
- You can find detailed information about these properties browsing at the following URL address: http://developer.android.com/reference/android/R.attr.html.

Theme Definition

- We can apply a theme by adding the android:theme attribute to the `<activity>` or `<application>` element in our android manifest file.

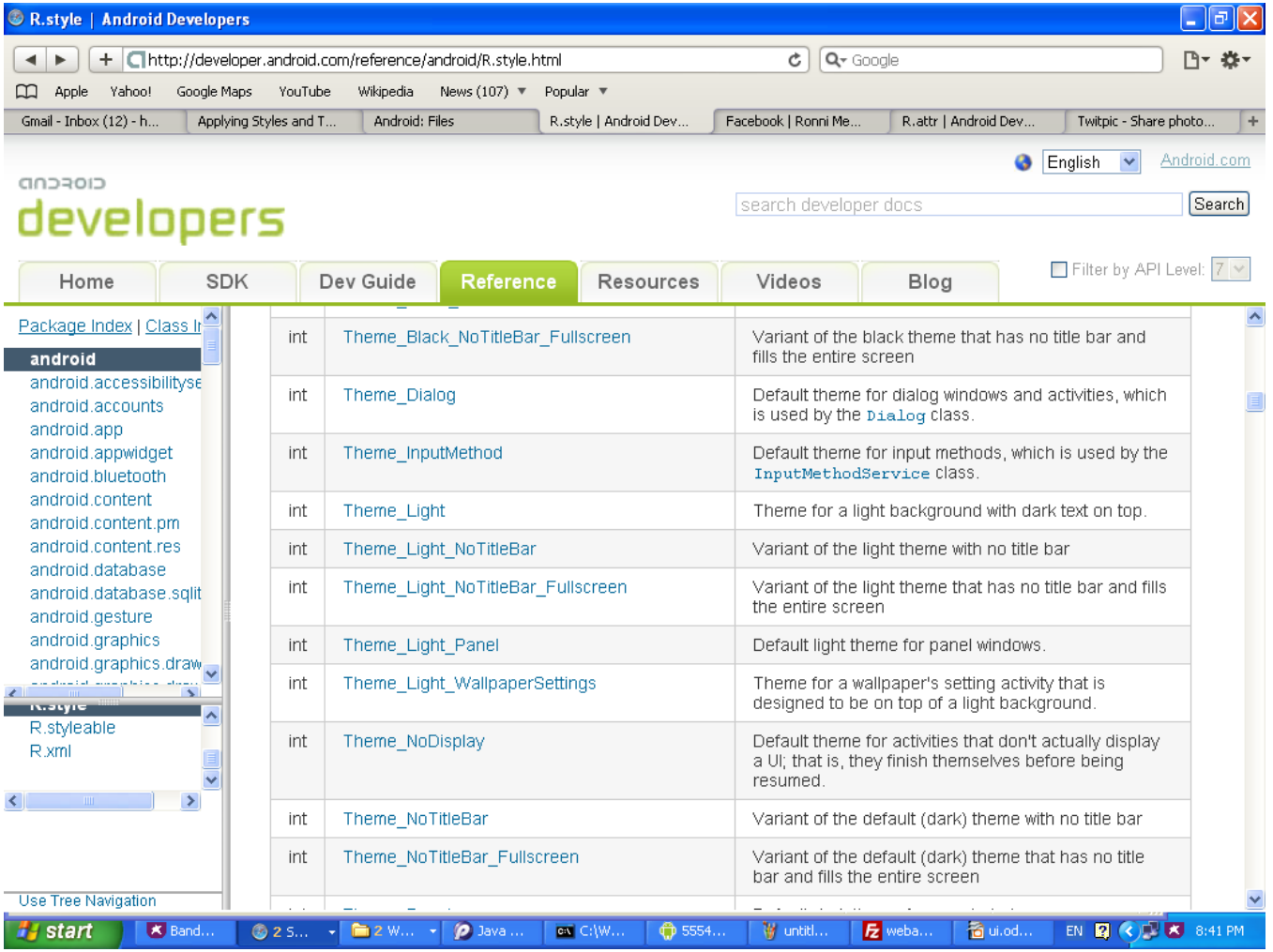
```
<application android:theme="@style/MyCustomTheme">
```

- Each and every view within the activity (or application) will apply each property it supports.

Predefined Themes & Styles

- There are various predefined themes you can choose to use in your activity (or application). You can find the complete list of predefined themes available browsing at http://developer.android.com/reference/android/R.style.html.

Predefined Themes & Styles



Predefined Themes & Styles

- When going over the predefined themes shown in the documentation you should change each '_' into '.'. For example, in order to refer `Theme_Light_Panel` you should refer `Theme.Light.Panel`.

Predefined Themes & Styles

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.abelski.samples"
      android:versionCode="1"
      android:versionName="1.0">
    <application   android:icon="@drawable/icon"
                   android:label="@string/app_name"
                   android:theme=
                    "@android:style/Theme.Light.NoTitleBar.Fullscreen">
        <activity android:name=".MySelectedThemeActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name=
                   "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="6" />
</manifest>
```

Predefined Themes & Styles

Inherit Predefined Theme

- It is always possible to inherit from a predefined theme changing one or more of the inherited properties.

Inherit Predefined Theme

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
    <style name="MyCustomTheme"
            parent="@android:style/Theme.Light.NoTitleBar.Fullscreen">
        <item name="android:windowBackground">@drawable/pixa</item>
    </style>
</resources>
```

Inherit Predefined Theme

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.abelski.samples"
      android:versionCode="1"
      android:versionName="1.0">

<application android:icon="@drawable/icon"
    android:label="@string/app_name"
    android:theme="@style/MyCustomTheme">

    <activity android:name=".ThemeActivity" android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

</application>

<uses-sdk android:minSdkVersion="4" />

</manifest>
```

Inherit Predefined Theme