آموزش برنامه نویسی اندروید

در محیط اندروید استودیو

متریال دیزاین

بخش دوم : آشنایی با Style و Theme در اندروید

مدرس : سید مهدی مطهری

www.android-studio.ir

در جلسه قبل با مفهوم کلی متریال دیزاین آشنا شدیم. یکی از موارد کلیدی در ساخت رابط کاربری، قابلیت تعریف Style (استایل) است.

استایل (Style) چیست؟

اگر با مفاهیم طراحی وب آشنایی دارید حتما با دیدن واژه استایل به یاد CSS می افتید. بله! در اندروید هم ما با همین ویژگی سروکار داریم. با این تفاوت که اینجا در قالب xml تعریف شده است.

فرض کنید در اپلیکیشن ما **۴** اکتیویتی وجود دارد که در هرکدام یک دکمه قرار داده ایم. می خواهم رنگ پس زمینه، رنگ و اندازه متن، فاصله متن دکمه از حاشیه و... در هر **۴** دکمه یکسان باشد. خب احتمالا اولین راهی که به ذهنتان می رسد این است که برای تک تک دکمه ها، این خواص را به صورت جداگانه درون هر اکتیویتی و داخل تگ مربوط به دکمه تعریف کنم. مشکلی نیست! اما وقتی به دردسر می افتم که تصمیم بگیرم در این دکمه ها تغییراتی ایجاد کنم. مثلا رنگ پس زمینه فعلی دکمه ها سبز است. حالا نظرم عوض شده و می خواهم آنرا به بنفش تغییر دهم. باید به سراغ تک تک اکتیویتی ها رفته و رنگ جدید را برای هر **۴** دکمه جایگزین مقدار قبلی کنم. علاوه بر اینکه وقت زیادی می گیرد ممکن است یک مورد را از قلم بیندازم و آنرا اصلاح نکنم. شاید برای 4 دکمه این اتلاف زمان محسوس نباشد یا احتمال خطا و فراموشی در حد پایینی باشد. اما در یک پروژه پیچیده و دارای صفحات و عناصر مختلف، باید آماده بروز این خطاها باشیم. پس لازم است از استایل بهره ببریم. در استایل یکبار ویژگی ها و خواص مدنظر را تعریف کرده و بی نهایت مرتبه در سراسر پروژه و اپلیکیشن خود آنرا بکار می بریم. با هر ایجاد تغییر در استایل، این تغییر در سراسر پروژه و در تمامی عناصر/ ویجت/ کنترل هایی که به آن ربط داده شده، اعمال خواهد شد.

تم (Theme) چیست؟

تم (قالب) همان استایل است که در کل اپلیکیشن بکار می رود. مثلا در تم تعریف می کنیم رنگ منوی اپلیکیشن (تولبار) خاکستری باشد. با افزودن این ویژگی به تم، رنگ موردنظر در تمام اکتیویتی ها اعمال شده و نیاز نیست استایل مدنظر را در هر اکتیویتی تعریف کنیم.

احتمالا تعاریف ذکر شده برای شما تا حدودی نامفهوم است. جای نگرانی نیست. یک پروژه ایجاد می کنم و به صورت عملی و در قالب مثال توضیح می دهم.

من یک پروژه با نام Style و MinSDK 16 ساختم. اگر بخاطر داشته باشید در بخش قبل اشاره شد که متریال دیزاین از اندروید ۵.۰ (Lollipop) اضافه شده و برای سازگاری نسخه های قبل از API 21 لازم است از کتابخانه appcompat-v7 استفاده کنیم. با توجه به اینکه پروژه ما حداقل API 16 را پشتیبانی می کند بنابراین به این کتابخانه نیاز داریم.

Build.gradle (app) را باز می کنم. بلاک مربوط به dependencies به اینصورت است:

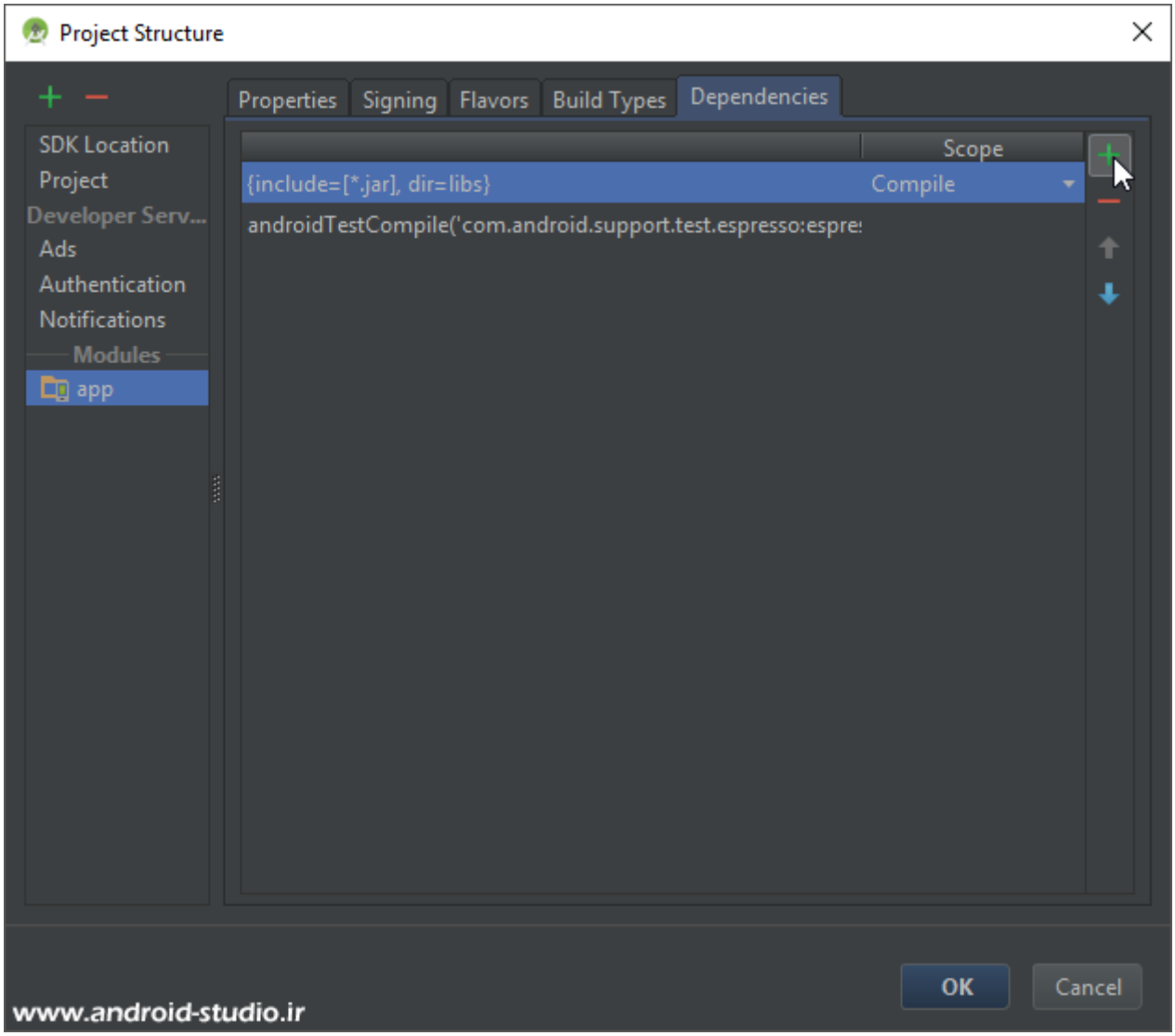
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:26.+'
}
```

بنابراین کتابخانه appcompat به صوت پیش فرض به پروژه من اضافه شده و لازم نیست کاری انجام دهم. اما اگر این کتابخانه در پروژه شما وجود نداشت باید به صورت دستی آن را اضافه کنید. راه ساده اضافه کردن خط

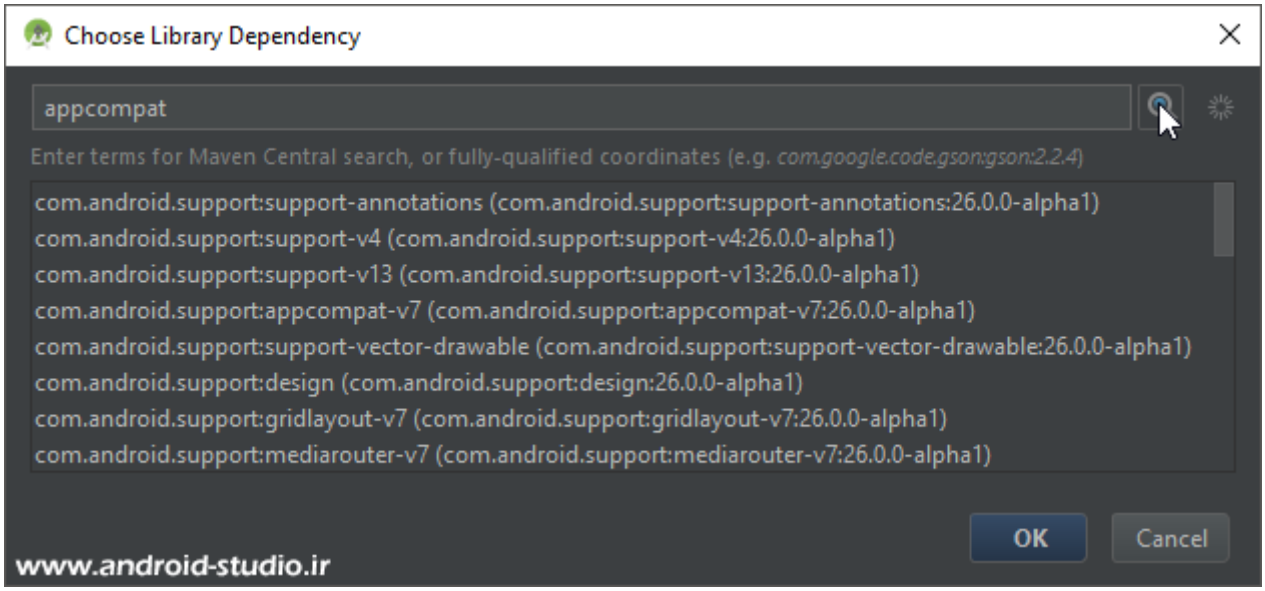
```
compile 'com.android.support:appcompat-v7:x.x'
```

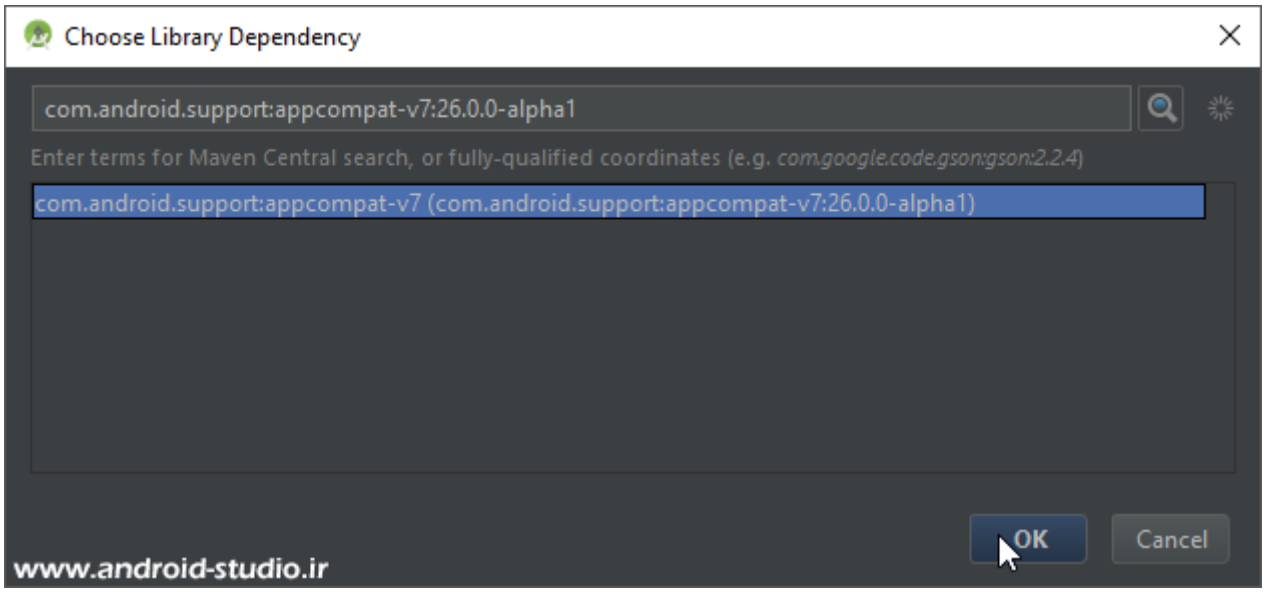
به پروژه است. اما ممکن است از ورژن appcompat موجود در SDK خود اطلاعی نداشته باشید (جایی که با x.x مشخص شده). بنابراین بهتر است از اندروید استودیو کمک بگیرید.

مسیر File > Project Structure سپس در پنجره باز شده در قسمت Modules گزینه app را انتخاب می کنم:



با انتخاب تب Dependencies لیست دپندنسی های موجود نمایش داده می شود. در سمت راست گزینه اضافه کردن (+) قرار دارد که با کلیک روی آن و انتخاب Library dependency پنجره جدیدی باز می شود. در اینجا لیست کتابخانه هایی که در SDK ما وجود دارد نمایش داده می شود. Appcompat را از لیست پیدا و یا در کادر جستجو وارد کنید:

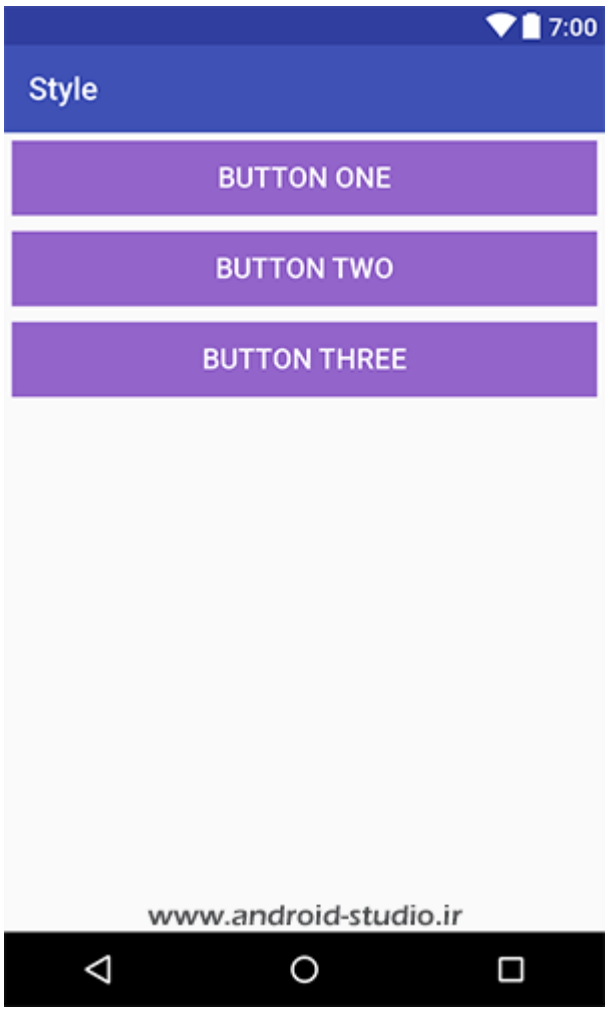




نکته: برای استفاده از کتابخانه appcompat لازم است Android Support Library را نصب کرده باشید.

با تایید پنجره فوق، کتابخانه به لیست Dependencies اضافه و در نهایت با تایید آن، به پروژه و فایل build.gradle اضافه می شود.

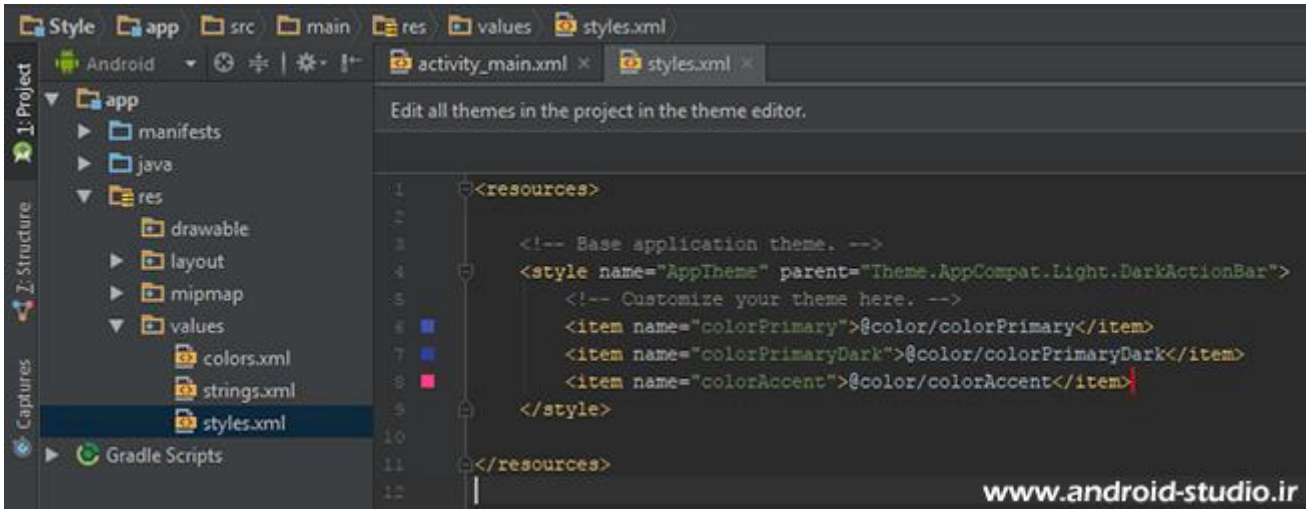
به سراغ اکتیویتی می روم. سه Button به activity_main.xml اضافه می کنم و خواص (attribute) یکسانی برای هرسه تعیین می کنم:



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="ir.android_studio.style.MainActivity">


    <Button
        android:id="@+id/button_one"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button One"
        android:textSize="18sp"
        android:textColor="#ffffff"
        android:background="#9365ca"
        android:layout_margin="5dp" />

    <Button
        android:id="@+id/button_two"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button Two"
        android:textSize="18sp"
        android:textColor="#ffffff"
        android:background="#9365ca"
        android:layout_margin="5dp"/>

    <Button
        android:id="@+id/button_three"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button Three"
        android:textSize="18sp"
        android:textColor="#ffffff"
        android:background="#9365ca"
        android:layout_margin="5dp"/>

</LinearLayout>
```

کد مربوط به دکمه ها را بررسی کنید. به جز id و text مابقی خواص یکسان است. این تکرار خواص علاوه بر افزایش حجم نهایی اپلیکیشن، روند اصلاح و تغییر را با مشکل مواجه می کند. الان اگر بخواهم مقدار مربوط به خاصیت background را تغییر دهم باید رنگ جدید را در هر سه مورد جایگزین کنم. حالا درنظر بگیرید در چند اکتیویتی دیگر هم ۱۰ دکمه دیگر داشته باشیم!

در مسیر res/values فایلی با نام styles.xml قرار دارد که مختص تعریف استایل هاست:



استایل ها داخل تگ <style></style> تعریف می شوند که برای هر استایل باید یک نام منحصر بفرد تعیین کنیم. هر یک از خواص داخل استایل نیز درون تگ <item></item> قرار می گیرند.

حالا می خواهم یک استایل جدید ایجاد کنم که خواص مشترک در دکمه ها را شامل شود. (استایلی که به صورت پیش فرض با نام AppTheme تعریف شده را فعلا نادیده بگیرید.)

```
<style name="MyButton">

</style>
```

یک استایل با نام **دلخواه** MyButton ایجاد کردم. در قدم بعد، خواص را در قالب item هایی به استایل اضافه می کنم.

فایل styles.xml:

```
<resources>

    <style name="MyButton">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textSize">18sp</item>
        <item name="android:textColor">#ffffff</item>
        <item name="android:background">#9365ca</item>
        <item name="android:layout_margin">5dp</item>
    </style>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

به اکتیویتی برمی گردم. ابتدا خواص مشترک در دکمه اول را حذف می کنم:

```
<Button
    android:id="@+id/button_one"
    android:text="Button One" />
```

حالا کافیست استایل MyButton را به این دکمه اضافه کنم:

```
<Button
    style="@style/MyButton"
    android:id="@+id/button_one"
    android:text="Button One" />
```

خاصیت style با مقدار @style/StyleName خواصی که در استایل تعریف کرده ام را روی این دکمه اعمال می کند.

```
<Button
    style="@style/MyButton"
    android:id="@+id/button_one"
    android:text="Button One" />

<Button
    android:id="@+id/button_two"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button Two"
    android:textSize="18sp"
    android:textColor="#ffffff"
    android:background="#9365ca"
    android:layout_margin="5dp"/>

<Button
    android:id="@+id/button_three"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Button Three"
    android:textSize="18sp"
    android:textColor="#ffffff"
    android:background="#9365ca"
    android:layout_margin="5dp"/>
```

در حال حاضر در قسمت Preview هرسه Button ظاهر یکسانی دارند. حالا دو دکمه دیگر را هم به استایل مرتبط می کنم:

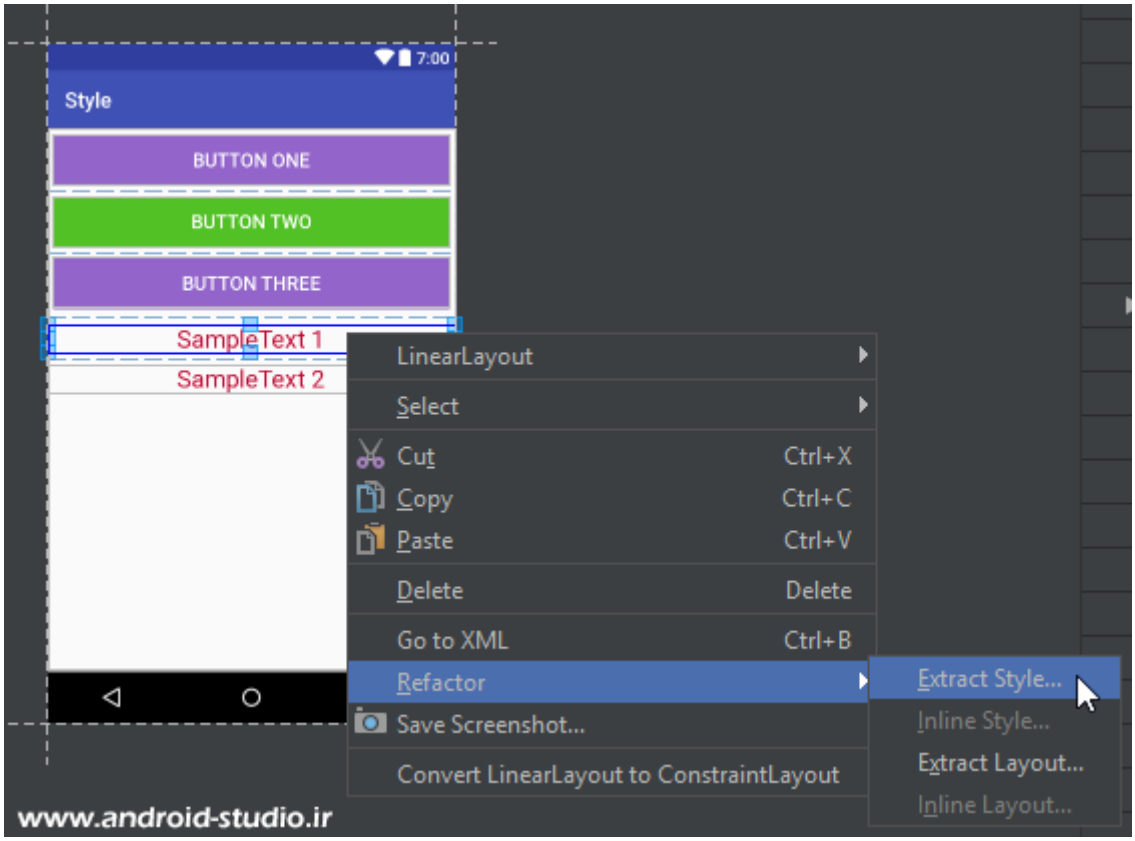
```
<Button
    style="@style/MyButton"
    android:id="@+id/button_one"
    android:text="Button One" />

<Button
    style="@style/MyButton"
    android:id="@+id/button_two"
    android:text="Button Two" />

<Button
    style="@style/MyButton"
    android:id="@+id/button_three"
    android:text="Button Three" />
```

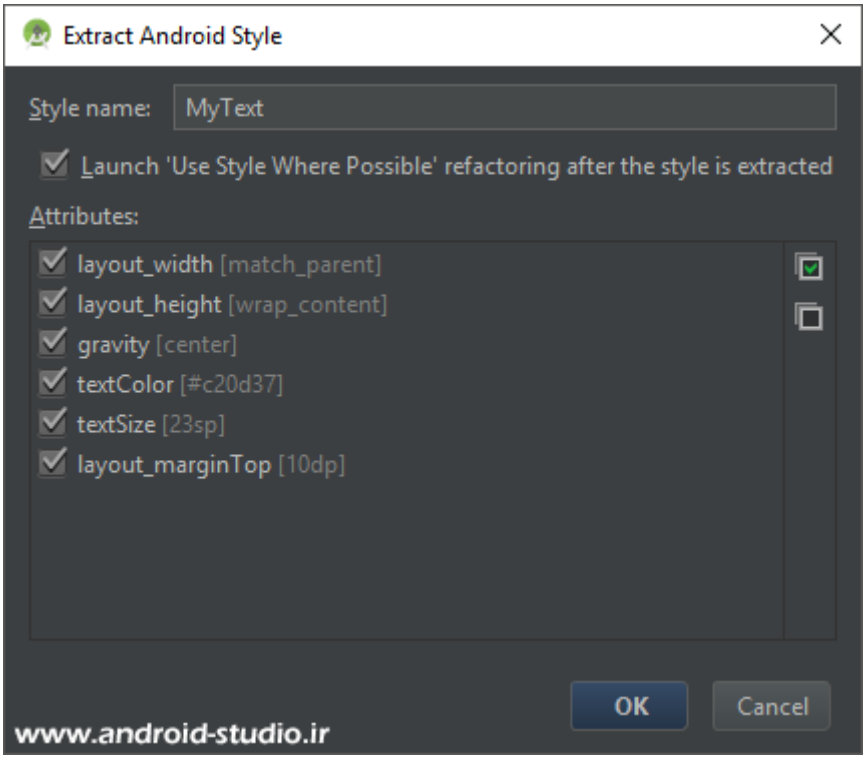
در ابتدا Layout اکتیویتی 40 خط کد داشت که با استفاده از استایل دهی به Button ها، این عدد به 25 خط رسید. علاوه بر تمیزی و کوتاهی کد نهایی، هرگاه نیاز به تغییری در خواص این دکمه ها داشته باشم، با یک بار تغییر آن در styles.xml، در تمامی سطح پروژه لحاظ خواهد شد و نگرانی از این بابت که موردی از قلم بیفتد ندارم.

در صورتی که مانند قسمت بالا قصد انتقال خواص موجود در یک کنترل به استایل را داشته باشیم، اندروید استودیو راه ساده تری را در اختیار ما قرار داده. من دو TextView با خواص مشترک به اکتیویتی اضافه می کنم تا دو قابلیت کاربردی را به شما نشان دهم.

```
<TextView
    android:id="@+id/txt_1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="SampleText 1"
    android:textColor="#c20d37"
    android:textSize="23sp"
    android:layout_marginTop="10dp"/>

<TextView
    android:id="@+id/txt_2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:text="SampleText 2"
    android:textColor="#c20d37"
    android:textSize="23sp"
    android:layout_marginTop="10dp"/>
```

در قسمت Preview روی یکی از TextView ها راست کلیک کرده و Refactor > Extract Style را انتخاب می کنم:

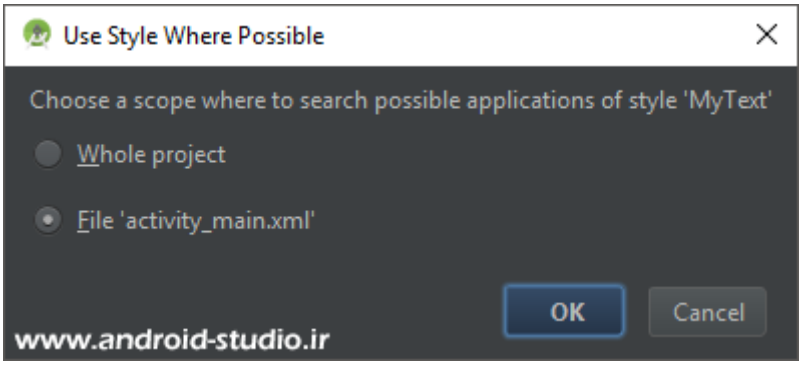




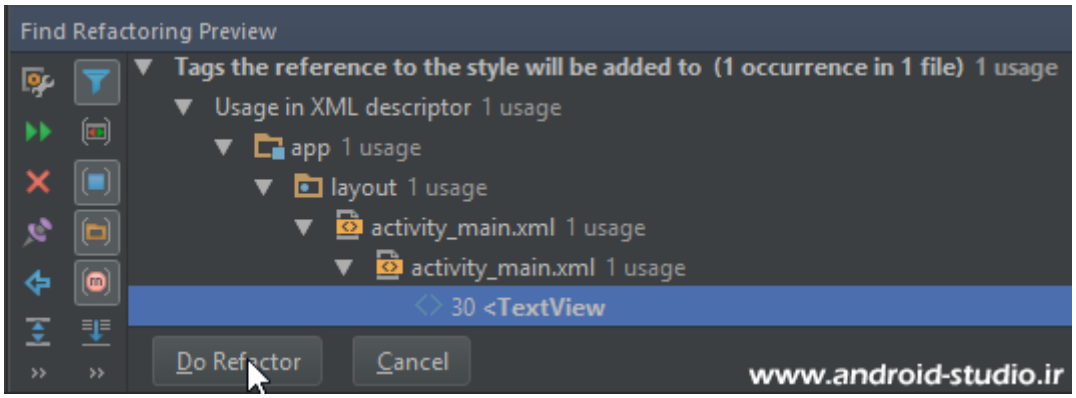
در پنجره جدید ابتدا یک نام برای استایل مدنظر انتخاب می کنم. در کادر Attributes خواصی که قصد دارم به استایل تبدیل شوند را باید انتخاب کنم. گزینه ای با عنوان

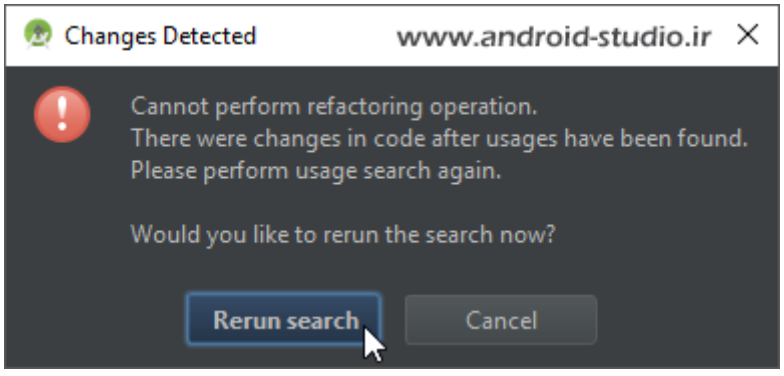
Launch ‘Use Style Where Possible’

وجود دارد که با انتخاب آن، تمام کنترل های موجود در پروژه (یا اکتیویتی) که خواص مشترک با مقادیر یکسان با این استایل داشته باشند، به صورت خودکار به آن مرتبط می شوند و نیاز به تبدیل دستی و تک به تک موارد نیست:



در این مرحله انتخاب می کنم که تغییرات برای سایر کنترل ها فقط در سطح همین اکتیویتی انجام شود یا کل پروژه که من فقط به همین اکتیویتی نیاز دارم.





در نهایت مجدد گزینه Do Refactor را تایید می کنم. استایل با موفقیت اضافه و روی هر دو TextView اعمال شد.

:styles.xml

```
<style name="MyText">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:gravity">center</item>
    <item name="android:textColor">#c20d37</item>
    <item name="android:textSize">23sp</item>
    <item name="android:layout_marginTop">10dp</item>
</style>
```

:main_activity.xml

```
<TextView
    android:id="@+id/txt_1"
    android:text="SampleText 1"
    style="@style/MyText" />

<TextView
    android:id="@+id/txt_2"
    android:text="SampleText 2"
    style="@style/MyText" />
```

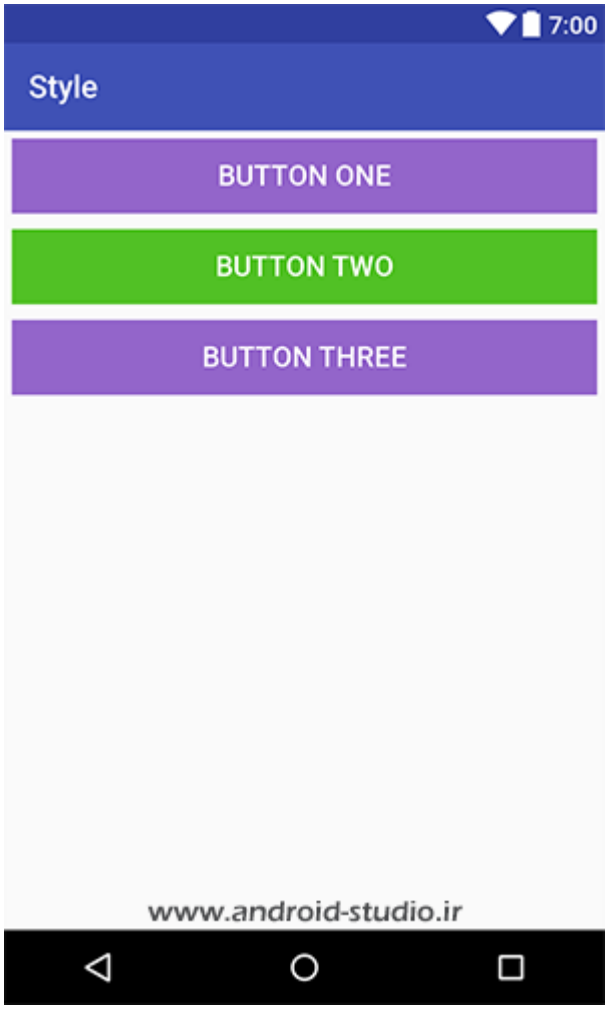
در استایل هم با مفهوم ارث بری سروکار داریم که کمک زیادی در کاهش حجم برنامه و افزایش سرعت توسعه برنامه می کند. به عنوان مثال من قصد دارم چند دکمه دیگر نیز در پروژه خودم بکار ببرم که ویژگی های آن با دکمه های قبل یکسان است و فقط در رنگ پس زمینه تفاوت دارد. اگر امکان ارث بری نداشته باشیم لازم است یک استایل جدید بسازم که تمام خواص استایل قبلی را تکرار کرده و فقط خاصیت background مقدار متفاوتی به خود گرفته. اما با استفاده از این قابلیت، تنها لازم است پس از ایجاد استایل جدید، خاصیت یا خواصی که قصد تغییر آنها را دارم بازتعریف کرده و مقدار مدنظر را جایگزین کنم.

```
<style name="MyButton">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:textSize">18sp</item>
    <item name="android:textColor">#ffffff</item>
    <item name="android:background">#9365ca</item>
    <item name="android:layout_margin">5dp</item>
</style>

<style name="MyButton.Green">
    <item name="android:background">#51c125</item>
</style>
```

به کد بالا دقت کنید. نام استایل جدید را MyButton.Green تعریف کردم که اضافه شدن .MyButton

قبل از Green باعث شده این استایل تمامی خواص استایل والد یعنی MyButton را نیز شامل شود. حالا خاصیت رنگ پس زمینه را با مقداری که مدنظر دارم به استایل اضافه می کنم. در نهایت این کار باعث می شود در هر ویجتی که MyButton.Green را به عنوان استایل تعریف میکنم، تمامی خواص MyButton را بپذیرد با این تفاوت که مقدار background در آن Override (جایگزین) شده است. برای اینکه ببینم استایل جدید من به درستی کار می کند یا نه، یکی از دکمه ها را به این استایل مرتبط می کنم:

```
<Button
    style="@style/MyButton"
    android:id="@+id/button_one"
    android:text="Button One" />

<Button
    style="@style/MyButton.Green"
    android:id="@+id/button_two"
    android:text="Button Two" />

<Button
    style="@style/MyButton"
    android:id="@+id/button_three"
    android:text="Button Three" />
```



مشاهده می کنید دکمه دوم تنها در رنگ پس زمینه متفاوت است. در نحوه ارث بری و نامگذاری استایل فوق مشکل خاصی متوجه ما نیست. اما بسته به نیاز توسعه دهنده ممکن است نامی که در این روش برای استایل جدید انتخاب می‌کنیم تا حدی طولانی شود که باعث شلوغی کدها خواهد شد. نام زیر را در نظر بگیرید:

MyButton.SmallButton.BorderButton.Green

- SmallButton: استایلی که در آن layout_width مقدار 100dp دارد و عرض را کامل اشغال نمی کند.
- BorderButton: استایلی که خاصیت مربوط به حاشیه به آن اضافه شده تا دکمه ها در حاشیه خود یک نوار رنگی داشته باشند.

در نهایت با ساخت استایلی با نام فوق، دکمه ای خواهیم داشت که سایز آن کوچک است، حاشیه دارد و رنگ پس زمینه سبز می باشد. این نام قدری طولانی است و بکار بردن آن در داخل اکتیویتی ها از زیبایی و خوانایی کد می کاهد. استایل مدنظر فوق را به صورت زیر می توانیم ارث بری کنیم:

```
<style name="GreenButton" parent="MyButton.SmallButton.BorderButton">

</style>
```

استایل هایی که لازم است ارث بری شوند در قالب parent تعریف می شوند و دیگر نیازی نیست به ابتدای نام اضافه شوند. یعنی به جای

```
<Button
    style="@style/MyButton.SmallButton.BorderButton.Green"
    android:id="@+id/button_one"
    android:text="Button One" />
```

به اینصورت استایل را به ویجت اضافه می کنیم:

```
<Button
    style="@style/GreenButton"
    android:id="@+id/button_one"
    android:text="Button One" />
```

خب! با کلیت Style در اندروید آشنا شدیم. حالا به معرفی Theme می پردازم. قبلا ذکر شد تم همان استایل است با این تفاوت که یکبار و در سطح کلی پروژه و اکتیویتی ها تعریف می شود نه در تعدادی ویجت خاص.

قبلا با فایل AndroidManifest.xml آشنا شدیم. تنظیمات کلی پروژه اندروید در این فایل تعیین می شود:

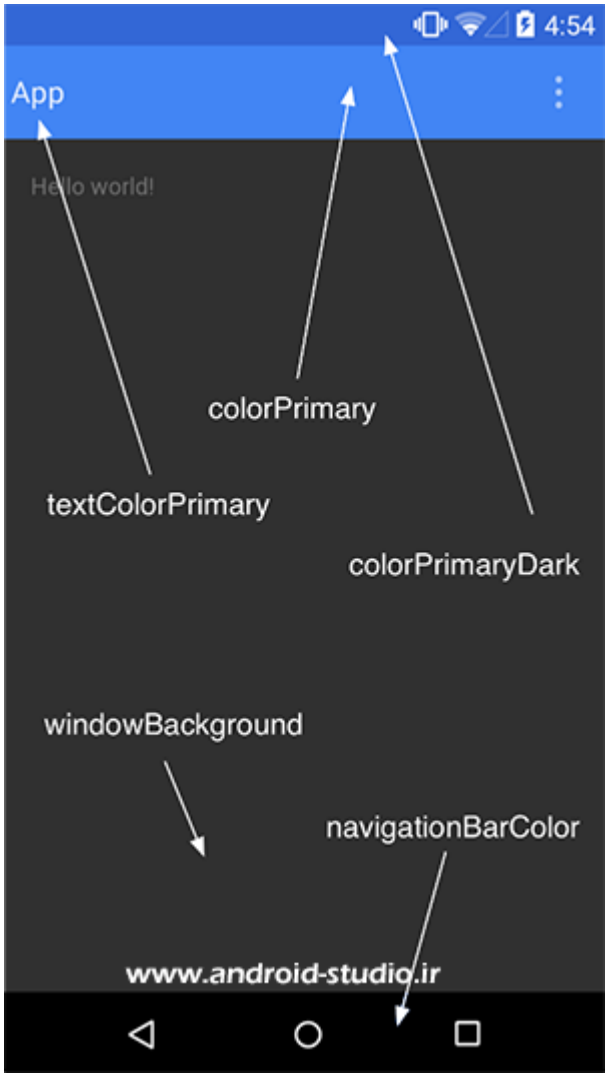
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ir.android_studio.style">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

درون تگ application خاصیتی با نام theme وجود دارد که مشابه آنچه قبلا دیدیم، به یک استایل با نام AppTheme مرتبط شده است. به styles.xml برگردید (با نگه داشتن ctrl و کلیک روی @style/AppTheme نیز به این فایل منتقل می شوید.

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

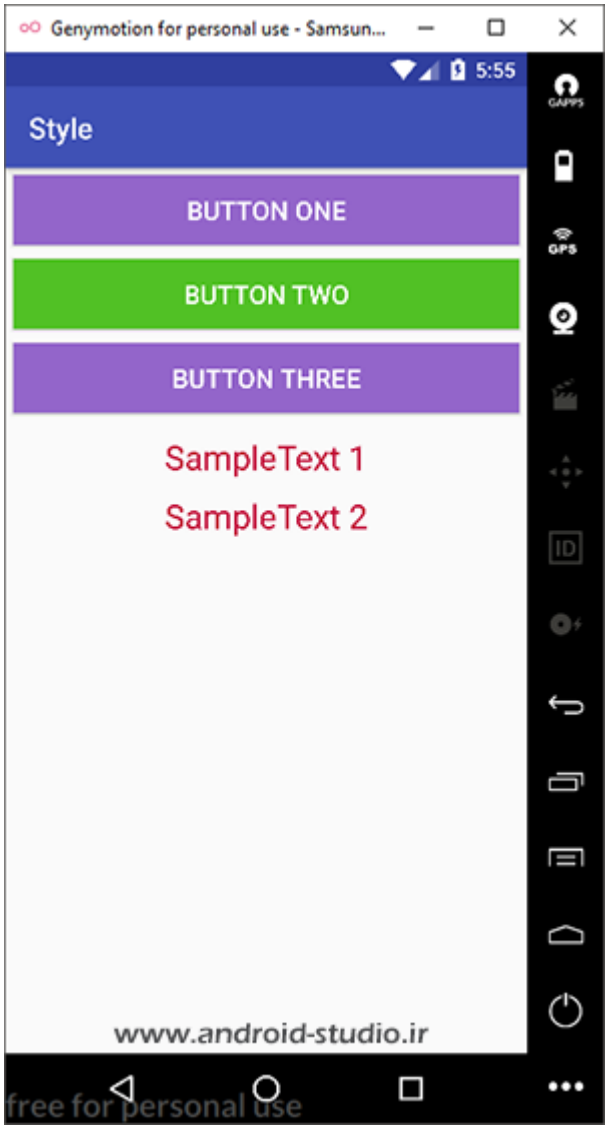
استایلی با نام AppTheme که از Theme.AppCompat.Light.DarkActionBar ارث بری شده است. با Theme.AppCompat ما به راحتی به تم های متریال که در کتابخانه appcompat-v7 گنجانده شده دسترسی داریم. با اضافه شدن Light، قسمت های مختلف تم (از جمله رنگ پس زمینه، نوشته، اکشن بار و...) به سبک لایت (روشن) تعیین شده اند که با حذف آن ظاهر تم از روشن به تاریک (Dark) تغییر می کند. تم متریال به صورت پیش فرض یک ActionBar دارد که رنگ آن روشن است و اگر DarkActionBar به تم اضافه کنیم به رنگ تیره تغییر می یابد.

رنگ بندی و تمامی attribute های پیش فرض تم را می توان تغییر داد. شاید این سوال برا شما مطرح شود که اگر این خواص قابل تغییر هستند پس چرا اندروید دو تم تاریک و روشن را به عنوان تم پایه معرفی کرده. پاسخ این است که توسعه دهنده با انتخاب نسخه ای که با تم نهایی مد نظرش همخوانی و هماهنگی بیشتری دارد، در جزئیات (مانند رنگ متن و آیکون های موجود در اکشن بار) نیاز به اعمال تغییرات کمتری خواهد داشت.



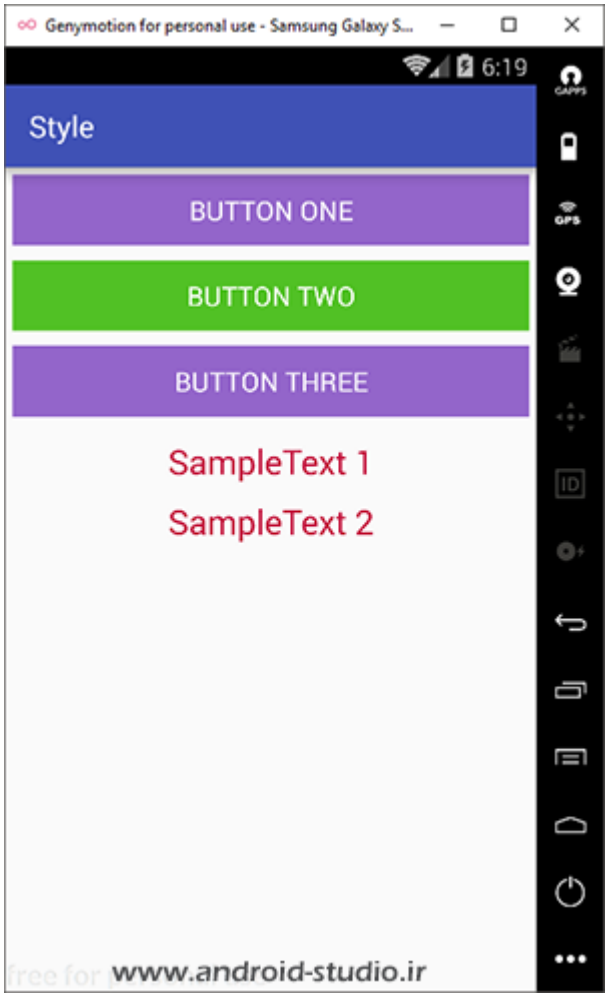
تم متریال خواصی را می پذیرد که از قبل تعریف شده است. به تصویر بالا دقت کنید. به عنوان مثال colorPrimaryDark مربوط به رنگ StatusBar (استاتوس بار) و colorPrimary مربوط به رنگ ActionBar می شود.

پروژه خودم را روی جنی موشن و اندروید نسخه ۷.۰ اجرا می کنم. با توجه به آنچه قبلا اشاره شد، قطعا این نسخه از اندروید، متریال دیزاین را کاملا پشتیبانی می کند:



یکی از ویژگی های متریال دیزاین این است که توسعه دهنده امکان تغییر رنگ پس زمینه استاتوس بار را دارد. همانطور که در تصویر بالا مشاهده می کنید بر اساس تم پیش فرض AppTheme در قسمت استاتوس بار رنگ سورمه ای پر رنگ و برای اکشن بار، سورمه ای کم رنگ لحاظ شده است. همچنین متن و آیکون در این دو قسمت به رنگ سفید نمایش داده می شود.

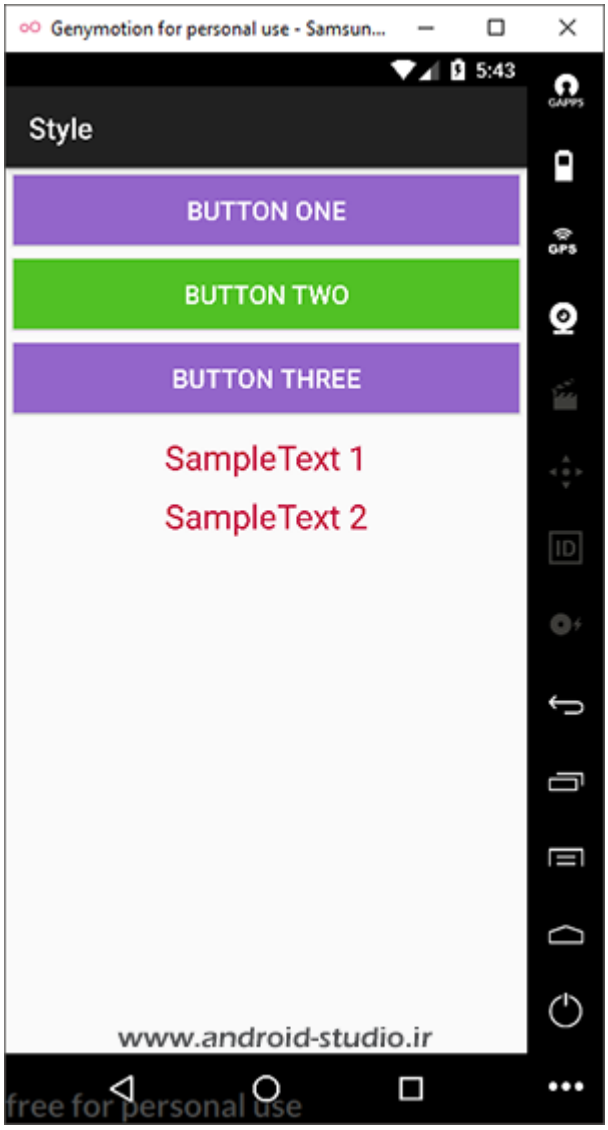
حالا پروژه را یکبار دیگر و بر روی نسخه ای از اندروید اجرا می کنم که پایین‌تر از API 21 باشد. یعنی نسخه ای قبل از آنکه متریال دیزاین معرفی شده باشد. من API 19 را انتخاب و اجرا می کنم:



همانطور که مشاهده می کنید در API 21 و به پایین، امکان تغییر رنگ استاتوس بار وجود ندارد و همواره با رنگ مشکی نمایش داده می شود. موارد دیگری نیز به اینصورت در نسخه های قبل از اندروید 5.0 پشتیبانی نمی شوند که لازم است توسعه دهنده تمامی قسمت های پروژه را در نسخه های مختلف اندروید تست و در صورت نیاز اصلاح کند.

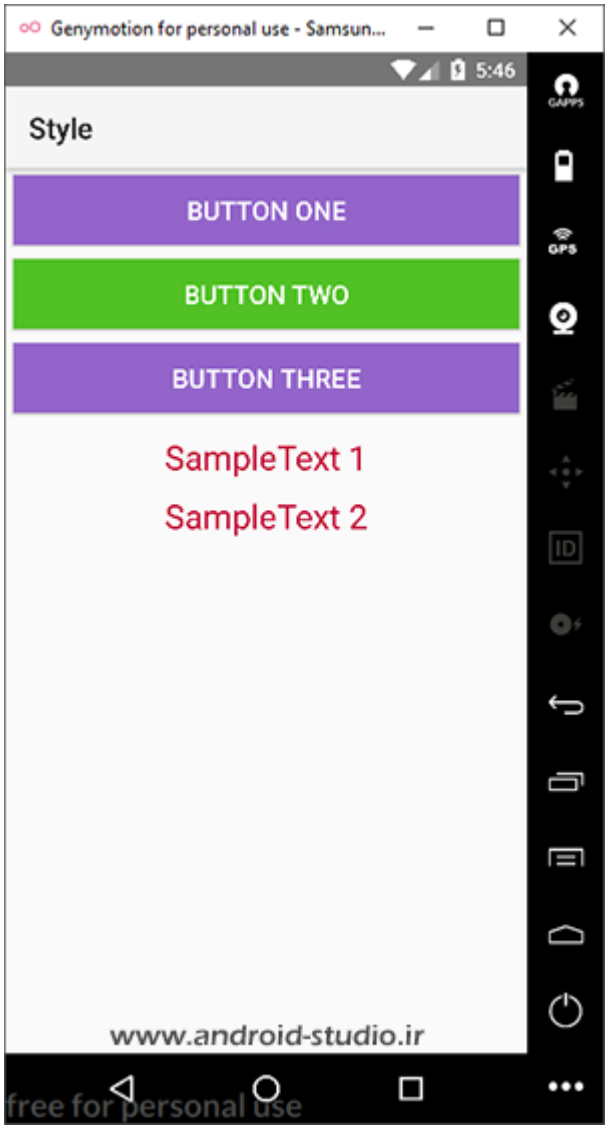
من خواص فعلی داخل AppTheme را حذف و سپس پروژه را مجدد Run می کنم:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
</style>
```



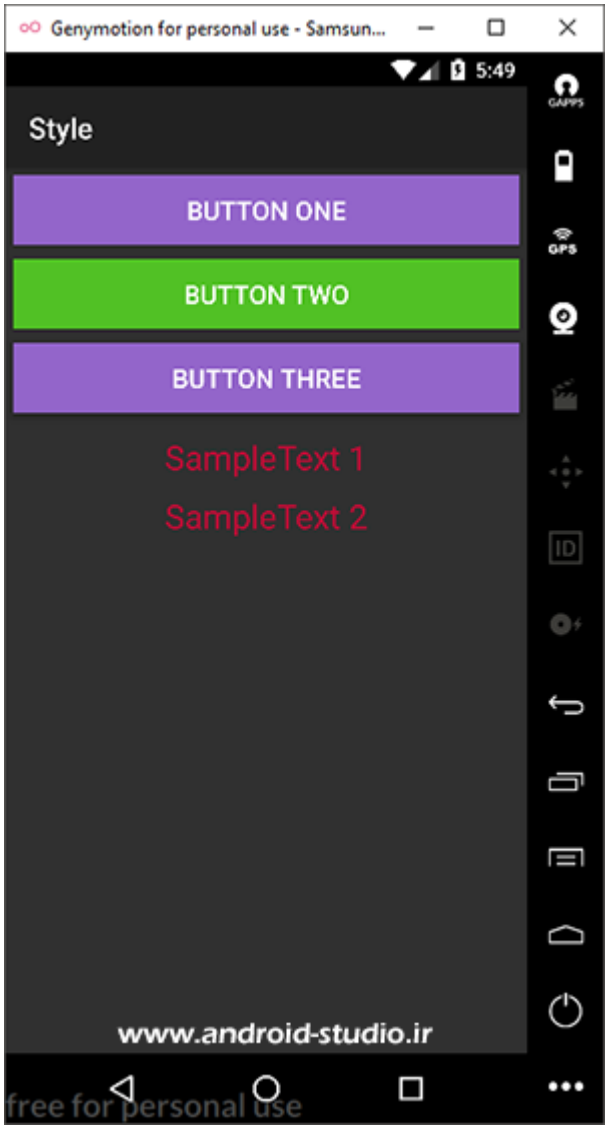
در تصویر بالا یک اکشن بار تاریک و استاتوس بار مشکی داریم که رنگ متن و آیکون در هر دو سفید است. در مرحله بعد DarkActionBar را حذف می کنم تا تم از آن ارث بری نکند:

```
<style name="AppTheme" parent="Theme.AppCompat.Light">

</style>
```



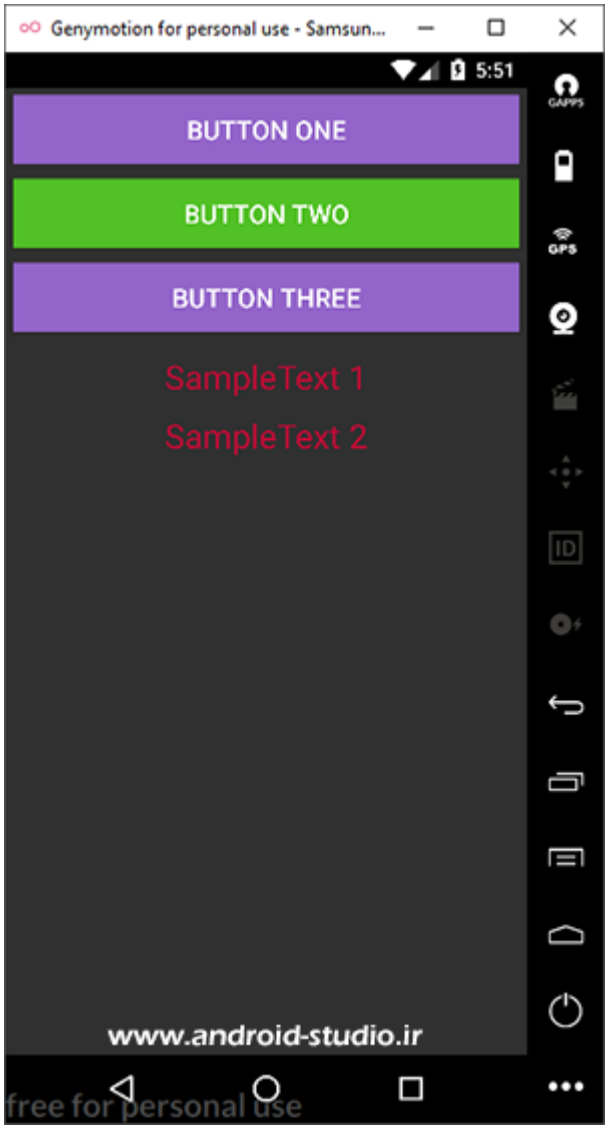
با حذف آن، یک اکشن بار روشن و استاتوس بار خاکستری رنگ در اختیار دارم که رنگ متن اکشن بار به مشکی تغییر یافته است. حالا Light را هم حذف و مجدد اجرا می کنم:

```
<style name="AppTheme" parent="Theme.AppCompat">

</style>
```



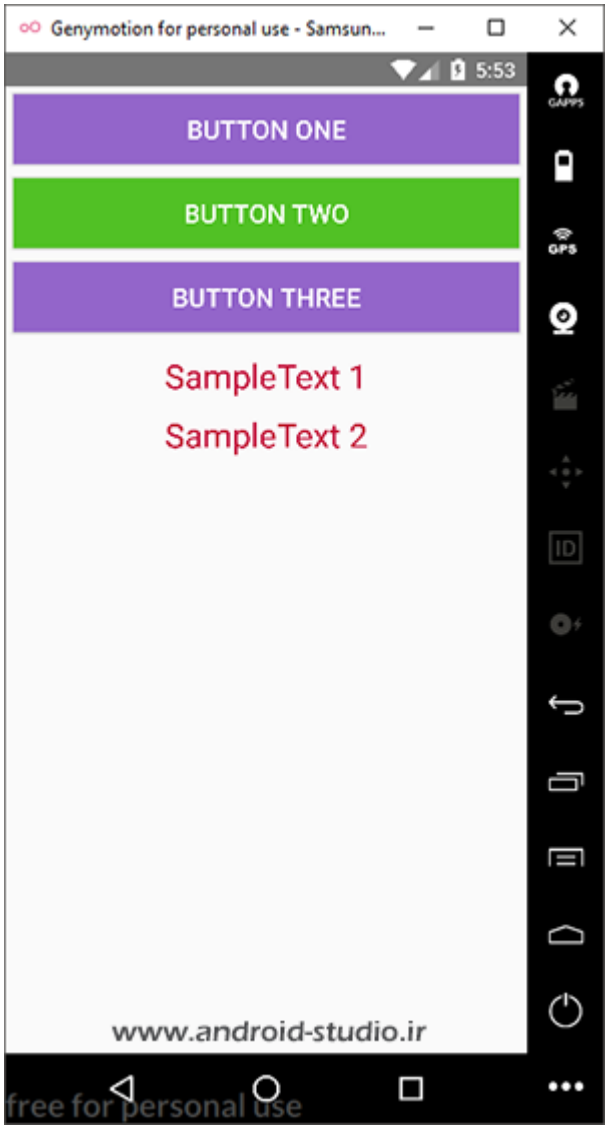
به دلیل اینکه تم متریال پایه در اندروید از نوع تاریک است، با حذف Light، سبک کلی تم نیز تاریک می شود. یعنی استاتوس بار و اکشن بار و همچنین پس زمینه اکتیویتی.

ممکن است در یک اپلیکیشن نیازی به اکشن بار نداشته باشیم. به راحتی و با جایگزین کردن NoActionBar به جای DarkActionBar اکشن بار از اکتیویتی ها حذف می شود:

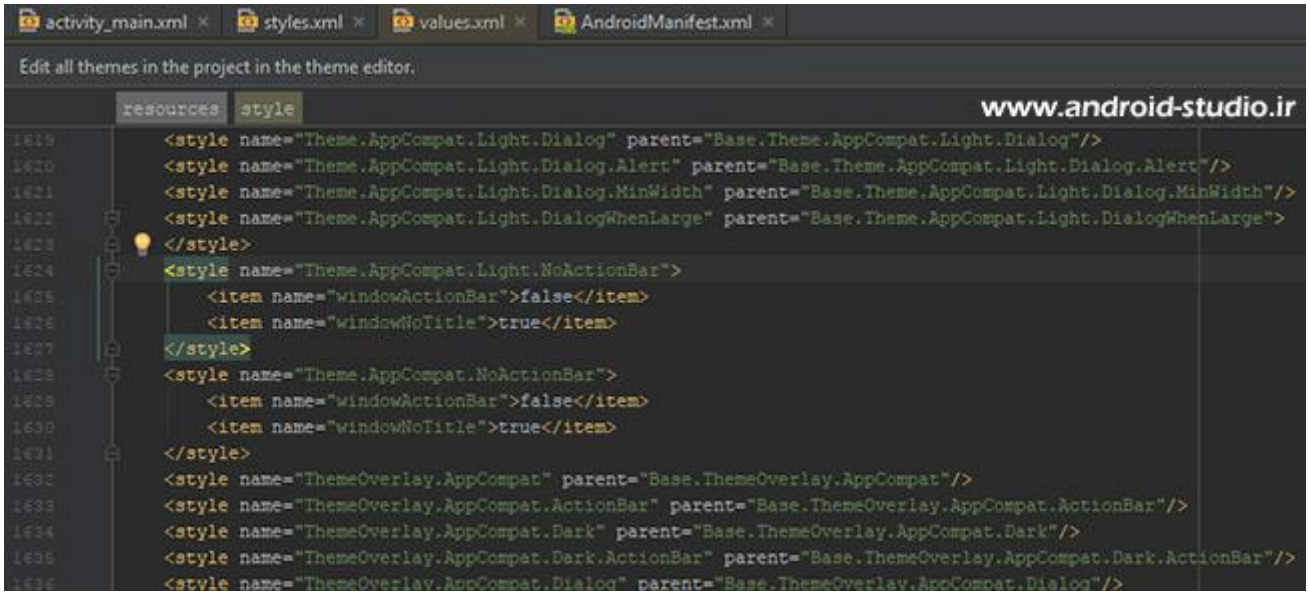


حالا اگر Light را مجدد به تم اضافه کنم، یک تم روشن بدون اکشن بار داریم:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

</style>
```

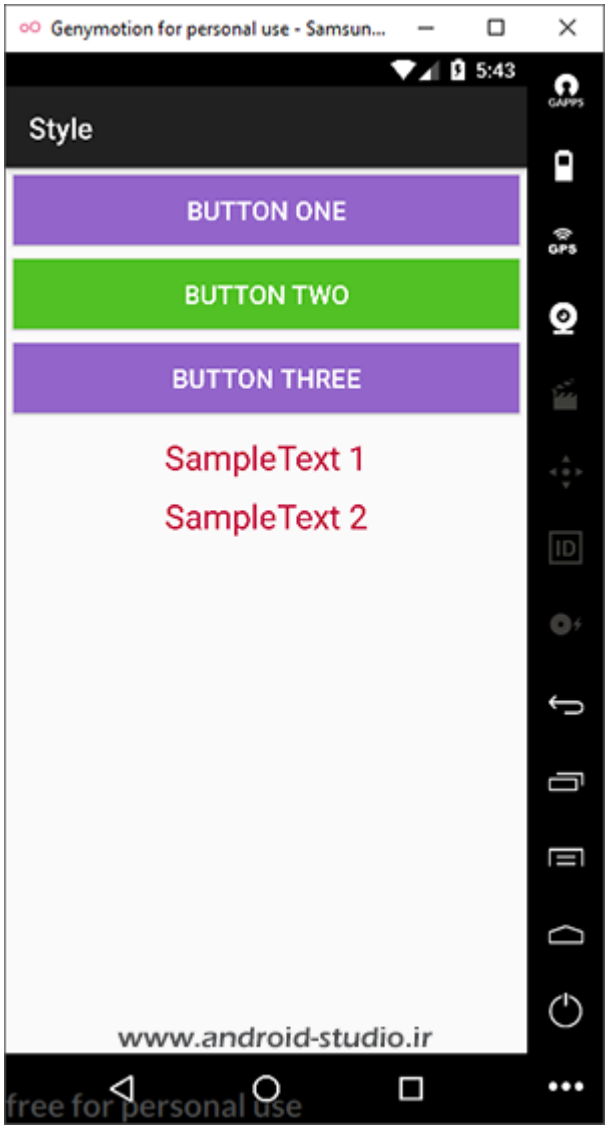


روی Theme.AppCompat.Light.NoActionBar کلید ctrl را نگه داشته، کلیک کنید. به فایلی منتقل می شوید که خواص مربوط به انواع تم ها در آن تعریف شده و به صورت پیش فرض استایل مربوط به NoActinBar را برای ما پیدا می کند:

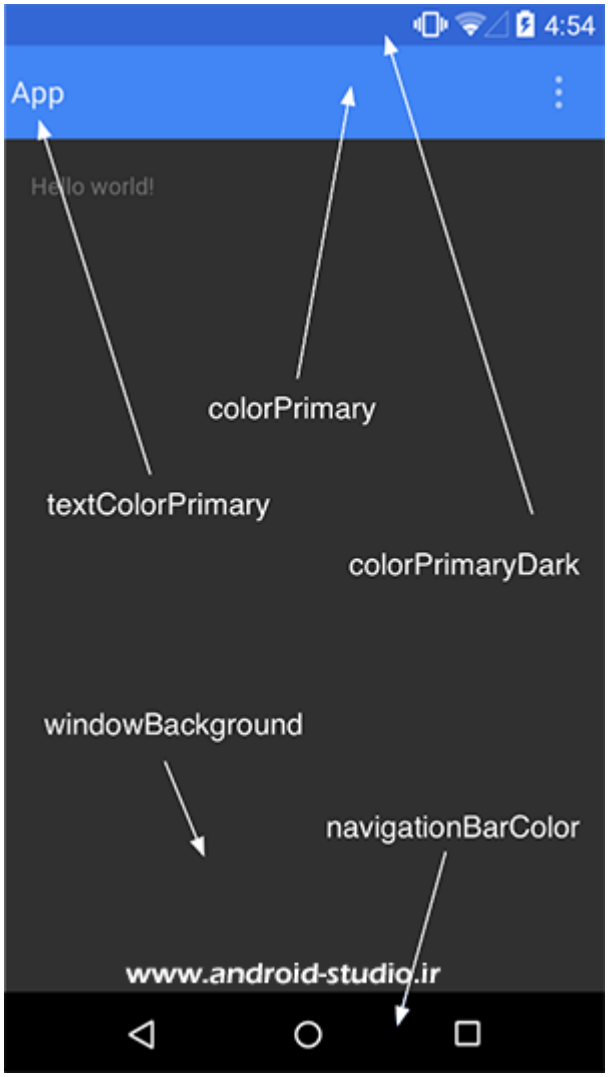


دو آیتم در این استایل قرار دارد. اولی windowActionBar با مقدار false که باعث حذف اکشن بار شده و دومی windowNoTitle که تایتل موجود در اکشن بار (یعنی کلمه Style) را نیز حذف می کند. اینجا ارث بری کار توسعه دهنده را ساده کرده و بجای اینکه لازم باشد در تم خود این دو خاصیت را تعریف کنیم، تنها با افزودن NoActionBar به Theme.AppCompat.Light این عمل انجام می شود. با بررسی سایر استایل های موجود در این فایل می توانید به تفاوتهای سایر تم ها با یکدیگر پی برده و از آنها استفاده کنید.

مجدد DarkActionBar را جایگزین NoActionBar می کنم تا اکشن بار به اکتیویتی اضافه شود. یعنی الان اپلیکیشن من به اینصورت است:

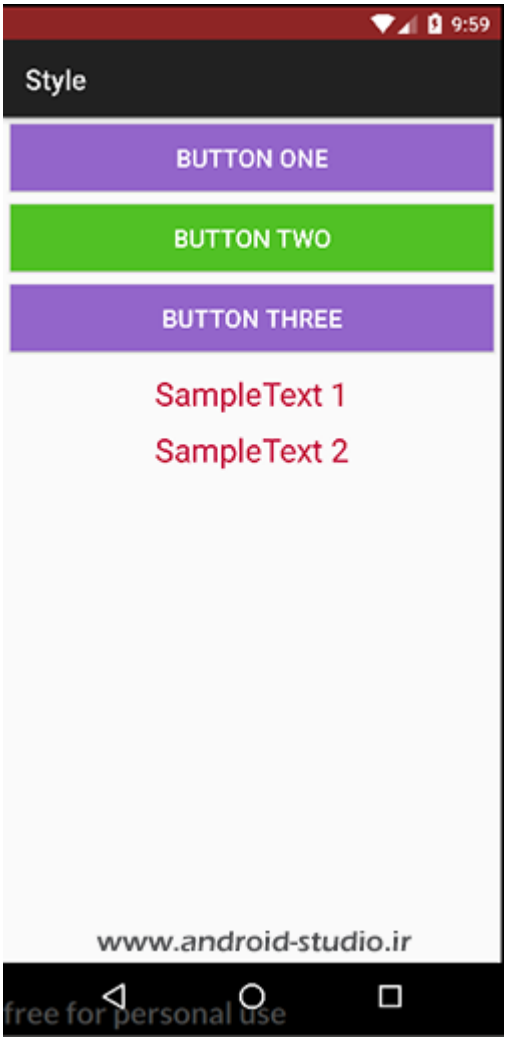


مجدد به تصویر زیر توجه کنید:



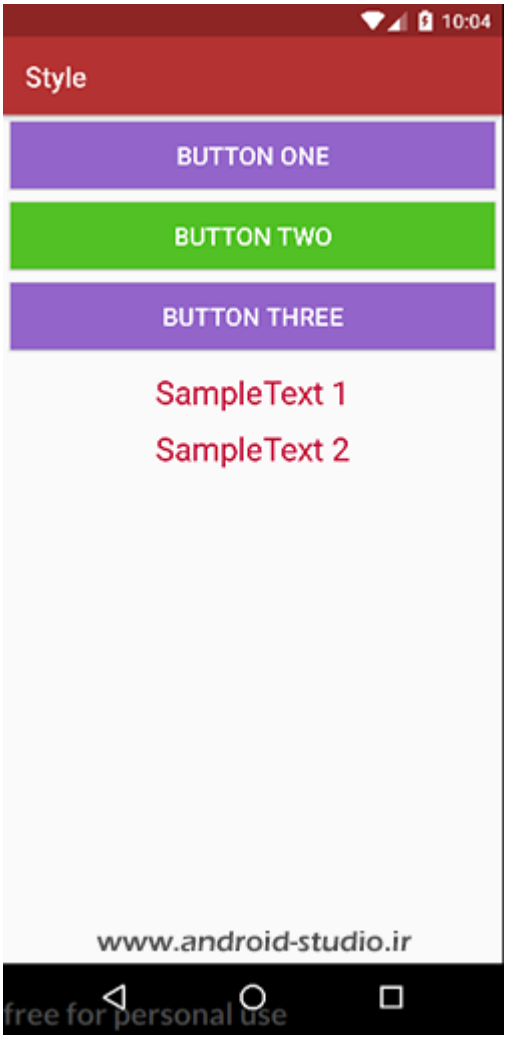
می خواهم مقدار رنگ استاتوس بار را با رنگ فعلی یعنی مشکی جایگزین کنم. یک آیتم با نام colorPrimaryDark و مقدار مدنظرم به تم اضافه می کنم:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimaryDark">#8e2525</item>
</style>
```



به همین ترتیب یک مقدار نیز برای colorPrimary در نظر می گیرم:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimaryDark">#8e2525</item>
    <item name="colorPrimary">#b43232</item>
</style>
```

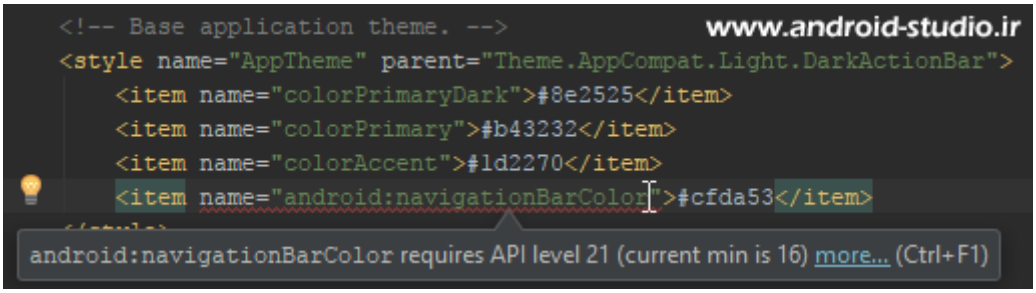


مورد بعدی که به تم اضافه می کنم colorAccent است. این رنگ در کنترل هایی مانند CheckBox، Switch، RadioButton، RatingBar، ProgressBar و... اعمال می شود. من دو مورد را به اکتیویتی اضافه کرده و کد رنگ #d22701 را نیز برای این خاصیت قرار می دهم:

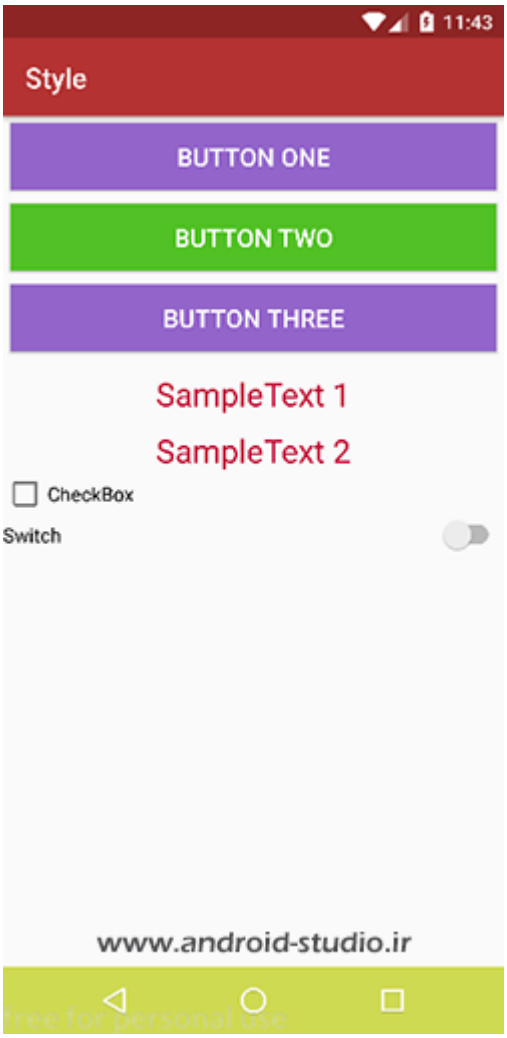
مشاهده می کنید رنگ چک باکس و سوئیچ در حالت فعال، همان رنگی است که به colorAccent نسبت داده ام.

در مرحله بعد تمایل دارم برای NavigationBar (نوار مشکی رنگ پایین صفحه که شامل 3 دکمه است) پس زمینه دیگری تعریف کنم:

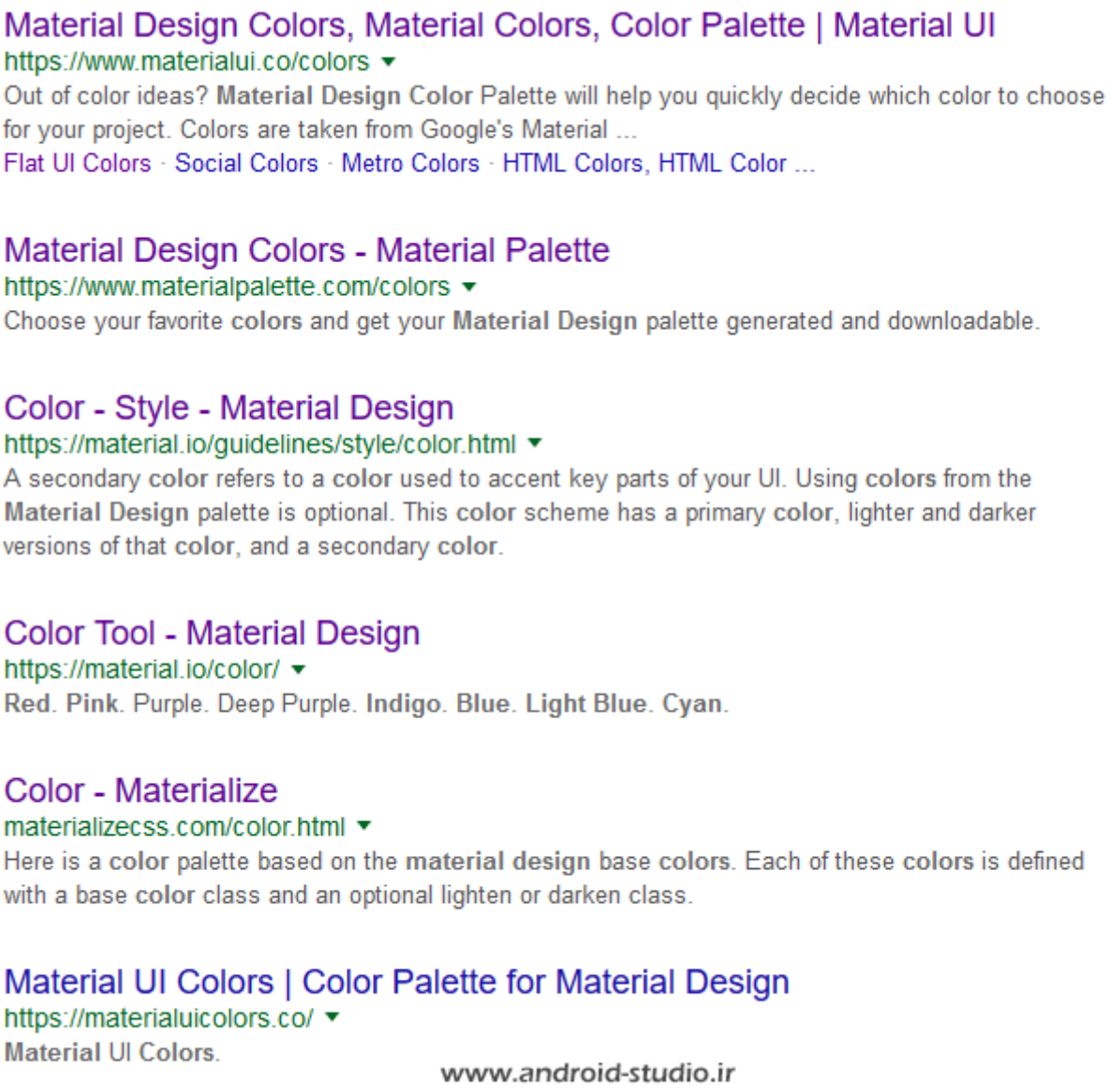
```
<!-- Base application theme. -->                    www.android-studio.ir
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimaryDark">#8e2525</item>
    <item name="colorPrimary">#b43232</item>
    <item name="colorAccent">#1d2270</item>
    <item name="navigationBar"
</style>          android:navigationBarColor
```



بر خلاف موارد پیشین، این مورد با پیشوند android: شروع شده که پس از تکمیل، اخطاری مطابق تصویر بالا نشان می دهد. به این معنی که این گزینه نیازمند حداقل API 21 است. خواصی که با android: شروع می شوند متعلق به appcompat نبوده و مربوط به خود اندروید می باشد. بنابراین این ویژگی فقط مربوط به نسخه هایی است که متریال دیزاین را به صورت بومی پشتیبانی می کنند:



اما رنگ هایی که من استفاده کردم برای متریال استاندارد نیست.

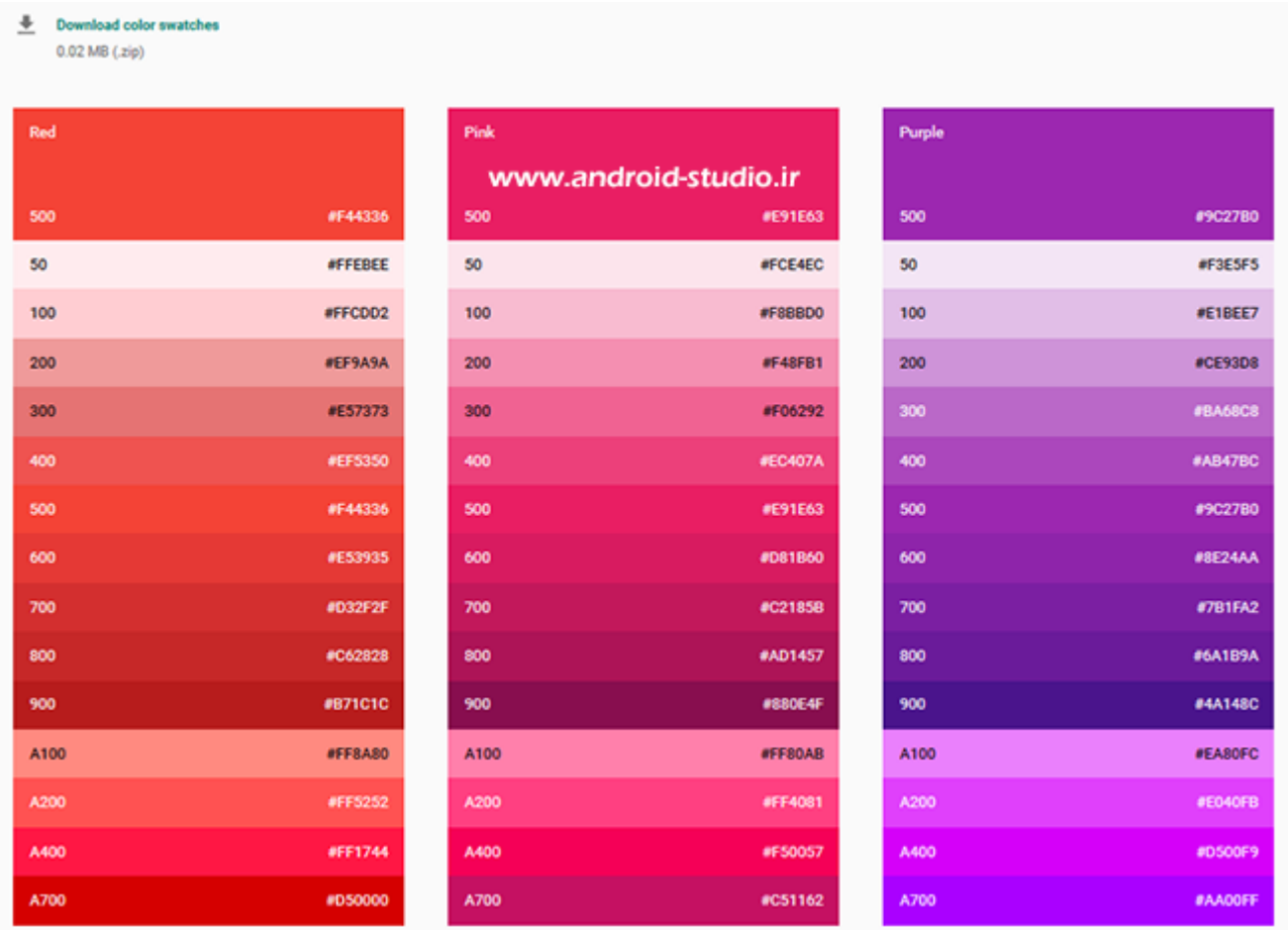


وب سایتهای زیادی در دسترس هستند که برای انتخاب رنگهای استاندارد به ما کمک می کنند. من Material design colors را سرچ کردم و چند وب سایتی که در صفحه اول نتایج معرفی می شود (تصویر بالا) در زمان تهیه این آموزش جزء منابع مطرح هستند.

Material.io متعلق به خود گوگل است و در خصوص متریال مطالب و مثالهای خوبی دارد که توصیه می کنم یکبار همه صفحات آن را مرور کنید. صفحه زیر مربوط به رنگهاست:

https://material.io/guidelines/style/color.html

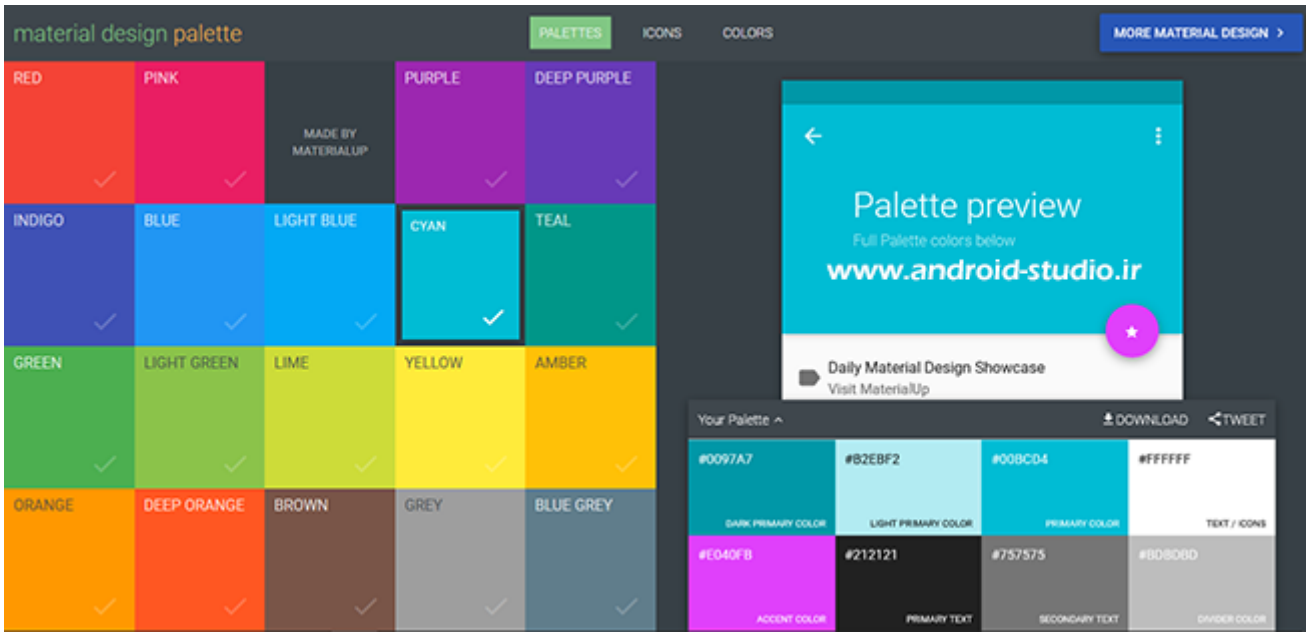
در قسمت Color Pallete تمامی رنگهای متریال با جزئیات ذکر شده است:



هر رنگ اصلی دارای چند زیرمجموعه با درصدهای مختلف است که می توان در نقاط مختلف از آنها استفاده کرد. به عنوان مثال عموما برای استاتوس بار از عدد ۷۰۰ و اکشن بار از عدد ۵۰۰ آن رنگ استفاده می کنند.

https://www.materialpalette.com

این وب سایت هم گزینه خوبی برای انتخاب رنگهای استاندارد است:



کافیست از داخل پالت، چند رنگ مدنظر خود را انتخاب کنید. در سمت راست هم یک دمو از صفحه اپلیکیشن فرضی را با همان رنگها مشاهده می کنید و هم کد رنگهای استفاده شده نمایش داده می شود که امکان دانلود با فرمت های مختلف نیز تعبیه شده.

موارد زیادی برای تم قابل تعریف است که فرصت بررسی همه آنها نیست. در هنگام افزودن آیتم جدید، لیستی باز می شود که تعداد زیادی از گزینه های در دسترس را می توان مشاهده کرد. علاوه بر این با بررسی مثالهای موجود در سطح وب هم در این زمینه اطلاعات بیشتری کسب خواهید کرد. ضمن اینکه این خواص محدود به رنگ نیست. به عنوان مثال با تعریف textSize، کلیه متون داخل پروژه که textSize اختصاصی ندارند، از طریق گزینه موجود در تم مقدار دهی خواهند شد.

اگر بخاطر داشته باشید آیتم هایی که به صورت پیش فرض درون AppTheme وجود داشت مستقیم به مقدار رنگ اشاره نداشتند:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

در اندروید فایلی با نام colors.xml داریم که وظیفه ای مشابه با strings.xml دارد که قبلا با آن آشنا شدیم. به جای تعریف مستقیم رنگ ها در استایل یا خواص ویجت ها، می توان ابتدا آنها را در colors.xml تعریف کرده سپس داخل پروژه استفاده کرد. به این ترتیب در مواقعی که نیاز به تغییر رنگ های داخل پروژه داریم، با سرعت بیشتری می توان تغییرات را اعمال کرد و در مواردی که قصد داریم در چند جای مختلف از یک رنگ یکسان استفاده کنیم، نیاز به تعریف چندباره نبوده و تنها با یک بار تعریف آن درون colors.xml می توان به تعداد نیاز آنرا در سراسر پروژه فراخوانی کرد.

نکته پایانی در خصوص تم اینکه قبلا با نحوه تعریف یک تم در مانیفست آشنا شدیم که با اضافه کردن تم به تگ application، در سراسر پروژه و تمامی اکتیویتی ها اعمال می شد. اما ممکن است در یک پروژه نیاز به استفاده از دو یا چند تم مختلف داشته باشیم. به عنوان مثال تصمیم دارم اکتیویتی اصلی دارای اکشن بار باشد اما اکتیویتی دوم که باید یک نقشه را نمایش دهد، بدون اکشن بار باشد. در اینجا لازم است تم مدنظر را ایجاد کرده و سپس داخل مانیفست و در تگ اکتیویتی مربوطه آن را اضافه کنیم.

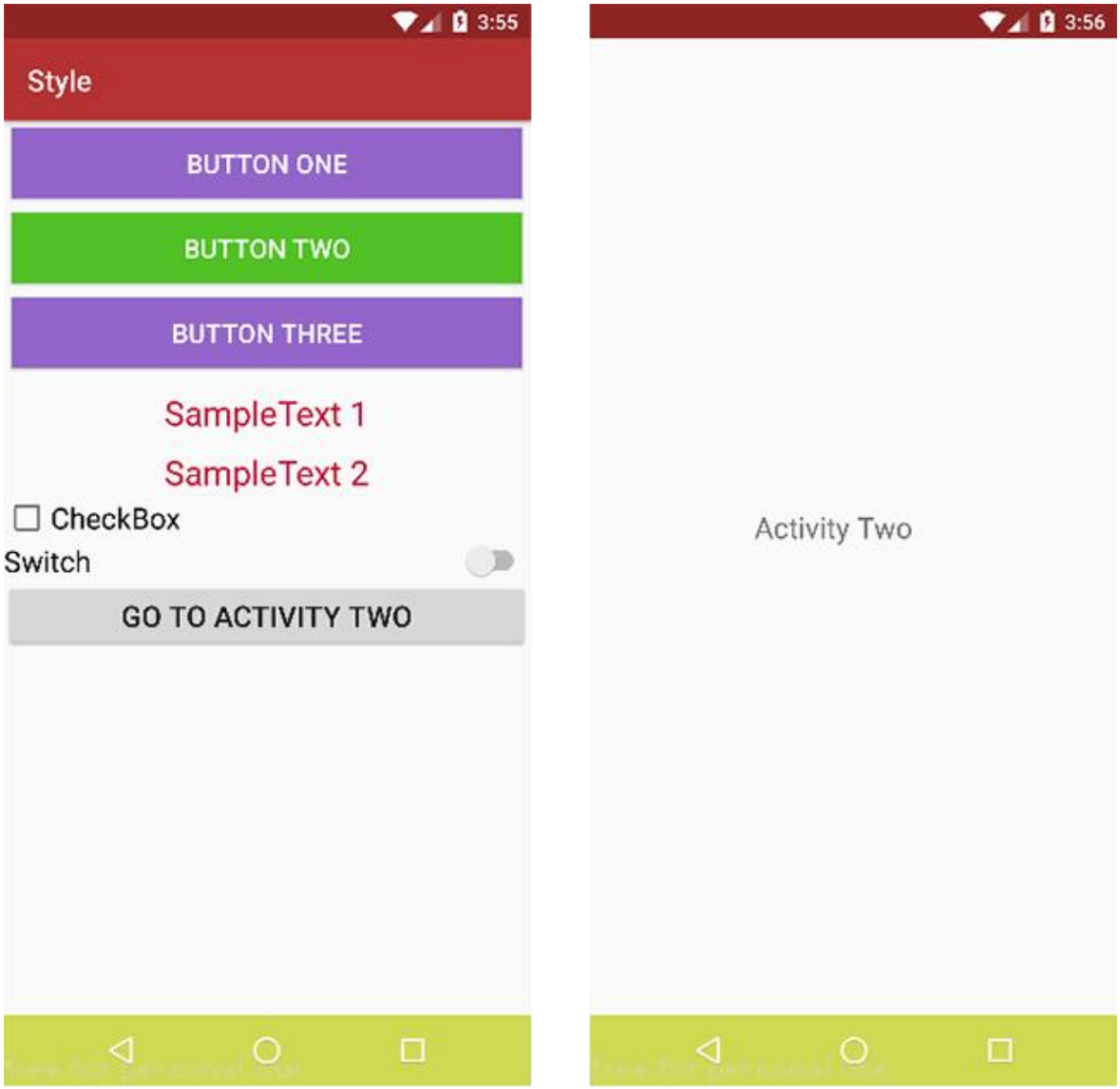
Styles.xml:

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimaryDark">#8e2525</item>
    <item name="colorPrimary">#b43232</item>
    <item name="colorAccent">#1d2270</item>
    <item name="android:navigationBarColor">#cfda53</item>
    <item name="android:textSize">20dp</item>
</style>

<style name="NoActionbarTheme" parent="AppTheme">
    <item name="windowActionBar">false</item>
    <item name="windowNoTitle">true</item>
</style>
```

:AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ir.android_studio.style">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".NoActionBarActivity"
            android:theme="@style/NoActionbarTheme">

        </activity>

    </application>

</manifest>
```

من یک تم جدید با نام NoActionBarTheme ایجاد و از AppTheme ارث بری کردم تا تمامی خواص آنرا دربر گیرد. سپس دو موردی که برای حذف اکشن بار لازم بود به تم اضافه نمودم. در قدم بعد یک اکتیویتی با نام NoActionBarActivity به پروژه اضافه کردم که به وسیله intent و با لمس دکمه Go to activity two از اکتیویتی اصلی به آن منتقل می شود و در نهایت تم را به تگ activity مربوط به این اکتیویتی اضافه کردم. هر اکتیویتی که تم اختصاصی نداشته باشد، از تم موجود در تگ application استفاده می کند:



www.android-studio.ir

کار با استایل و تم نیازمند تمرین، آزمون و خطا و مشاهده مثالهای بیشتری است و هرچه پروژه های بیشتری انجام دهید، نکات جدیدی خواهید آموخت.

توجه: سورس پروژه درون پوشه Exercises قرار داده شده است

با ارائه انتقادات و پیشنهادات خود، ما را در ارائه آموزش های بهتر یاری فرمائید.

www.android-studio.ir