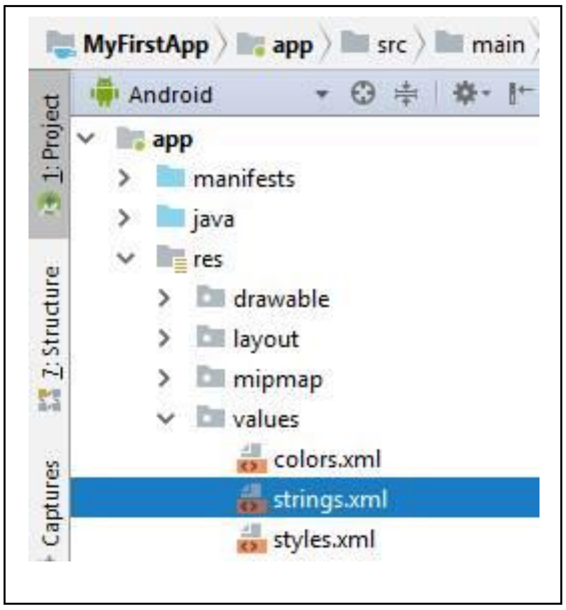
Zarqa Uni.
Internet Technology
Android Programming
Dr. Khaled Alqawasmi

# Resources

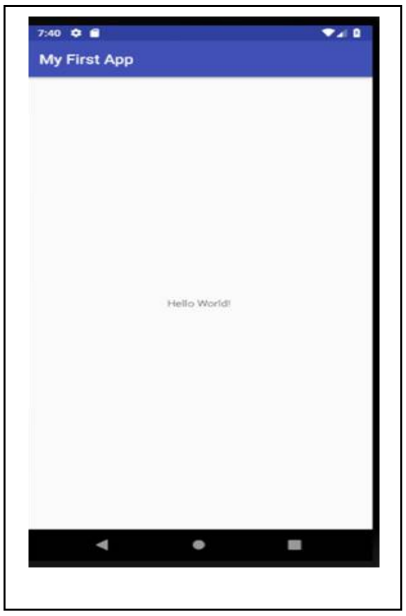
## 1) String resources

- String resources are defined in the `strings.xml` file in the `res/values` folder



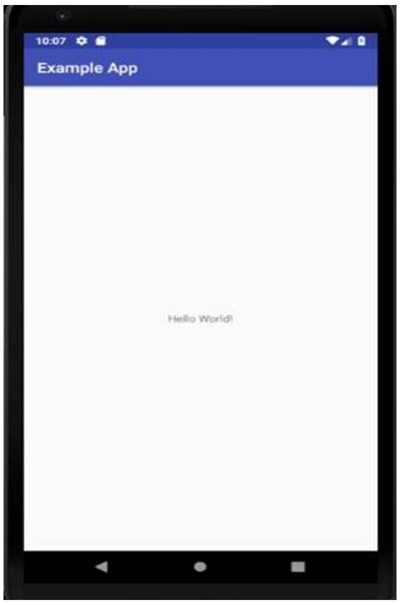
- Defining strings an app uses in this file is more effective than in source code
- Changes to a string are made in one central location separate from the source code
- As a simple example, let’s make a change to a string in our example app and view the result
- Open `strings.xml` and note the string defined in the variable `app_name`

```
<resources>
    <string name="app_name">My First App</string>
</resources>
```



- We can easily edit this string in the XML file to change the name of our app
- We never modify any source code associated with the app, only the XML file

```xml
<resources>
    <string name="app_name">Example App</string>
</resources>
```

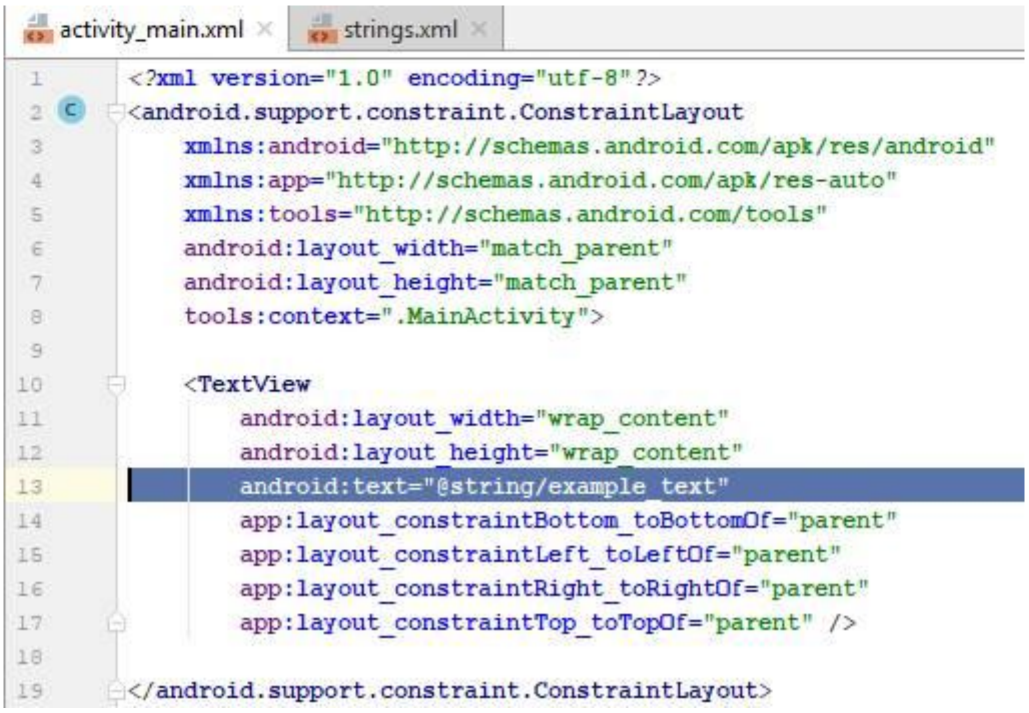


we can add new string as follow:

```xml
<resources>
    <string name="app_name">Example App</string>
    <string name="example_text">This is an example text</string>
</resources>
```

- Note the new string resource also reflected in the `activity_main.xml`
- The `@string` is used to indicate that an XML string variable is used as the text

Zarqa Uni.
Internet Technology
Android Programming
Dr. Khaled Alqawasmi


activity_main.xml strings.xml

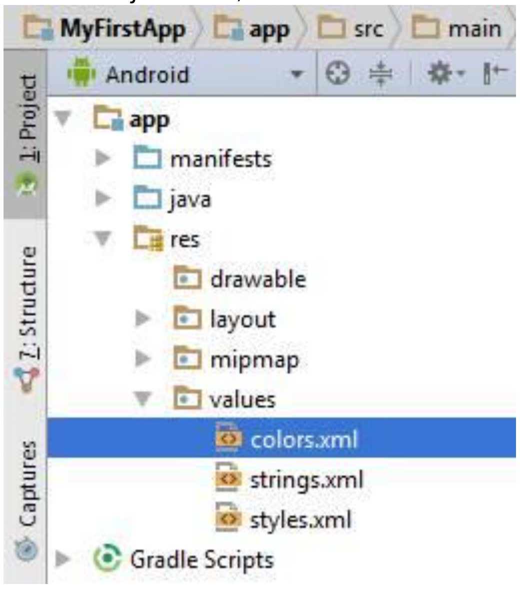
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/example_text"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

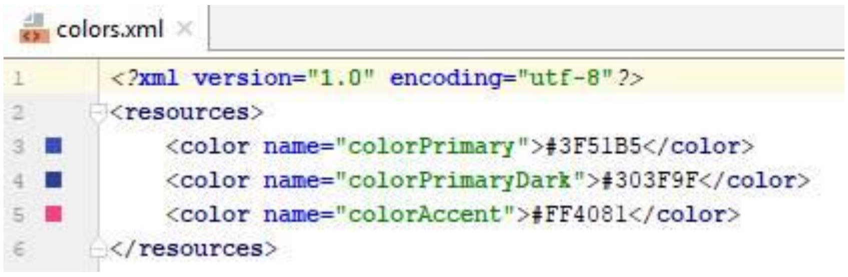
## 2) **color resources**

- We can also create and use XML resources that specify colors
- Let's add an XML resource to represent a color we can use for our text
- We'll create this resource manually as we did with our string resources
- In our *Project* view, locate the `colors.xml` file under `res/values` directory



- Double-click this file to display the contents of this file as shown below

Zarqa Uni.
Internet Technology
Android Programming
Dr. Khaled Alqawasmi



colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

- Before investigating colors, let's discuss color represented in hexadecimal format

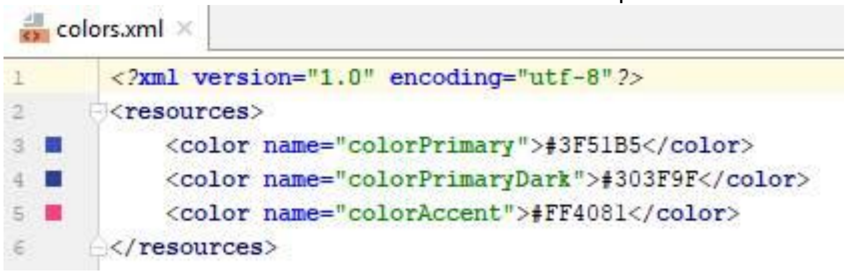
## Hexadecimal colors

- Android provides XML specification of colors using hexadecimal digits
- The android site contains information on specifying such color values
- Each hexadecimal digit represents 4 bits and two hex digits represent a byte
- Colors can be specified in different bit resolutions

| | |
|---|---|
| - #AARRGGBB | 32 bit color specification w/ 8 bit alpha, red, green, blue |
| - #RRGGBB | 24 bit color specification w/ 8 bit red, green, blue (alpha is opaque) |
| - #ARGB | 16 bit color specification w/ 4 bit alpha, red, green, blue |
| - #RGB | 12 bit color specification w/ 4 bit red, green, blue (alpha is opaque) |

- In 24-bit color, 8 bits are reserved to specify each component
- For example, black is 000000, medium-gray is 808080, white is ffffff
- In 12-bit color, 4 bits are reserved to specify each component
- For example, black is 000, medium-gray is 888, white is fff
- Any conversion table between decimal-hex-binary can assist selecting values
- Any visual color conversion tool can assist with color selection

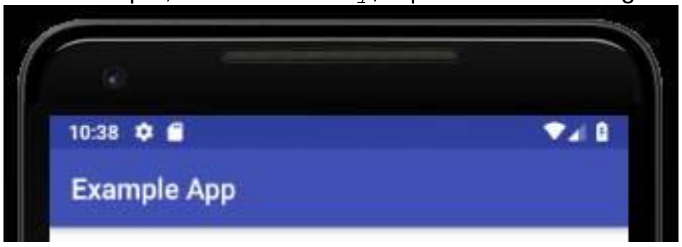
- Android Studio provides a utility to edit our hexadecimal color resources
- Let's examine the colors in our colors.xml file repeated below



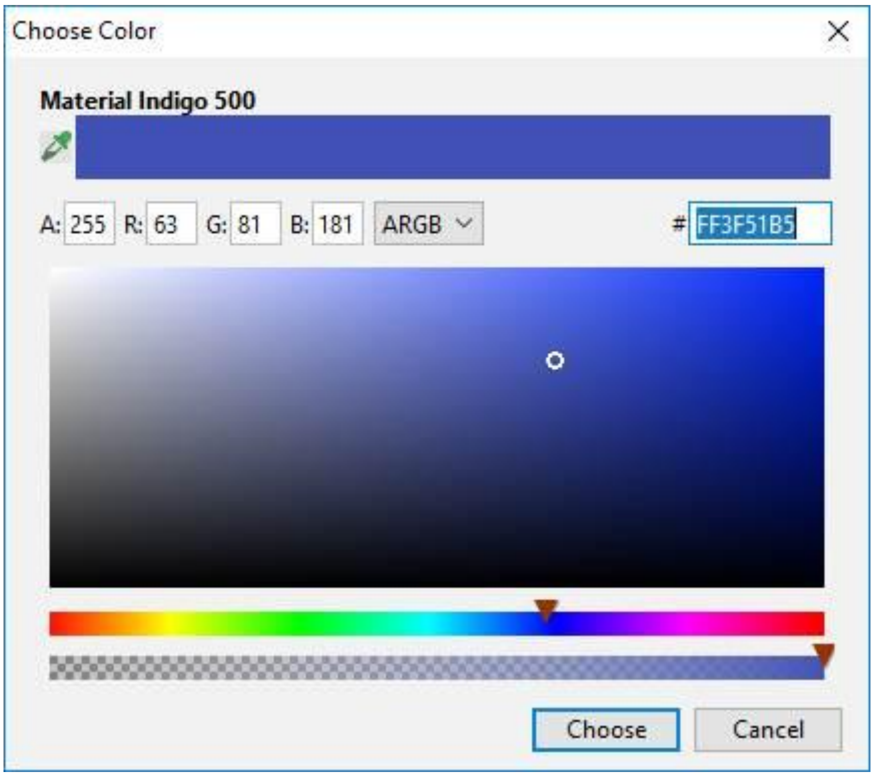
colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

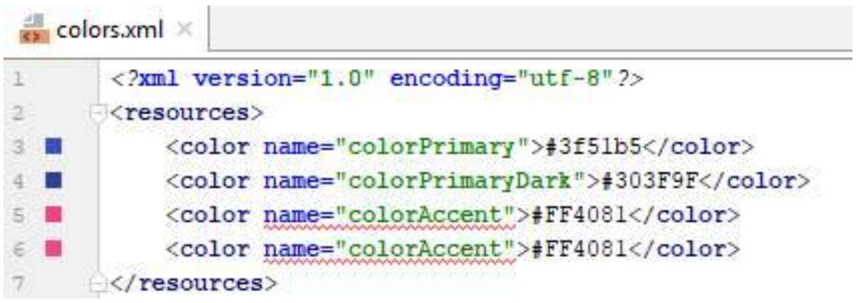
- These are colors that are currently used in our example first app
- For example, colorPrimary, represents the background color under the app name



- Note the squares to the left of the XML color resources in colors.xml
- Clicking on a square opens a selection tool to edit the color

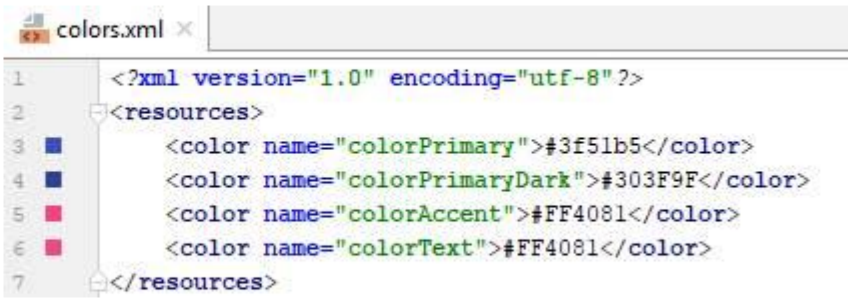


- To create a new color, we can simply copy and paste the last color in `colors.xml`



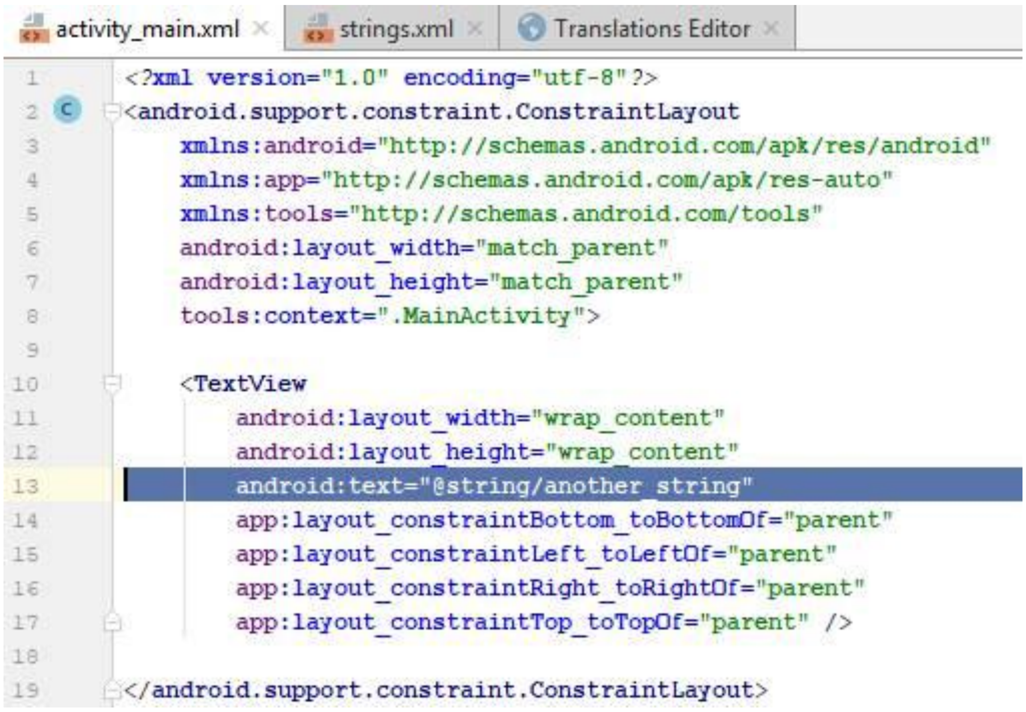
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3f51b5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

- Next, we'll simply rename the XML variable name to `colorText` as shown below



```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3f51b5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorText">#FF4081</color>
</resources>
```

Let's apply this new color resource as the color of our string displayed in the Activity
- Below is the string specified in the `TextView` object in `activity_main.xml`
- Note: click in the `android:text` field if you don't see the string variable name

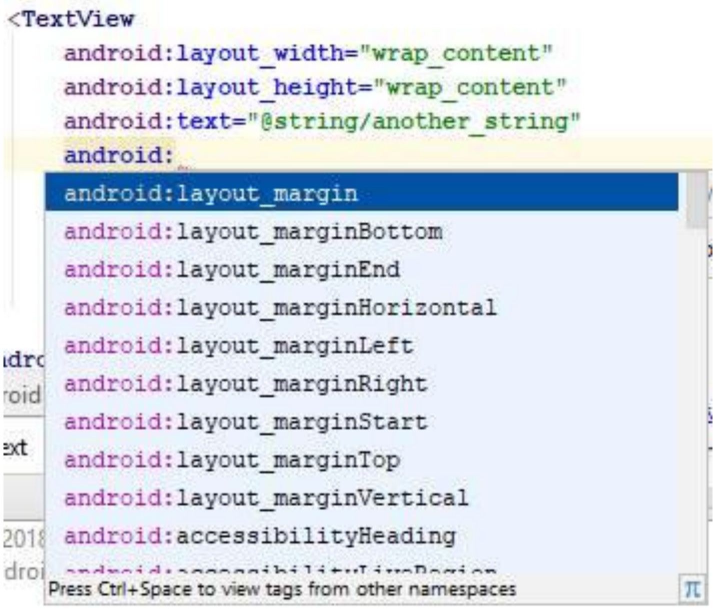


```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/another_string"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

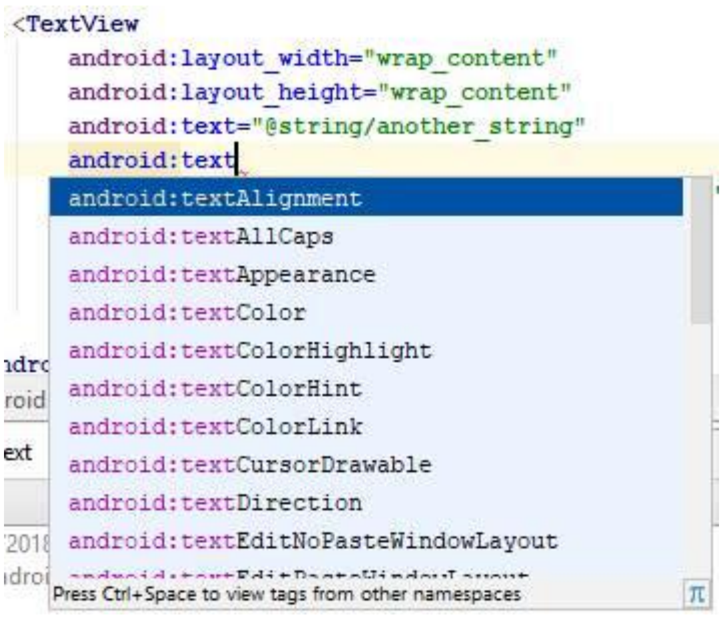
- One way to change the string text color is by manually editing this XML file
- Enter a new line under the `android:text` field
- On this new line type `android:` then type Ctrl-Space
- Android Studio offers fields to complete the XML specification



```xml
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/another_string"
    android:
```

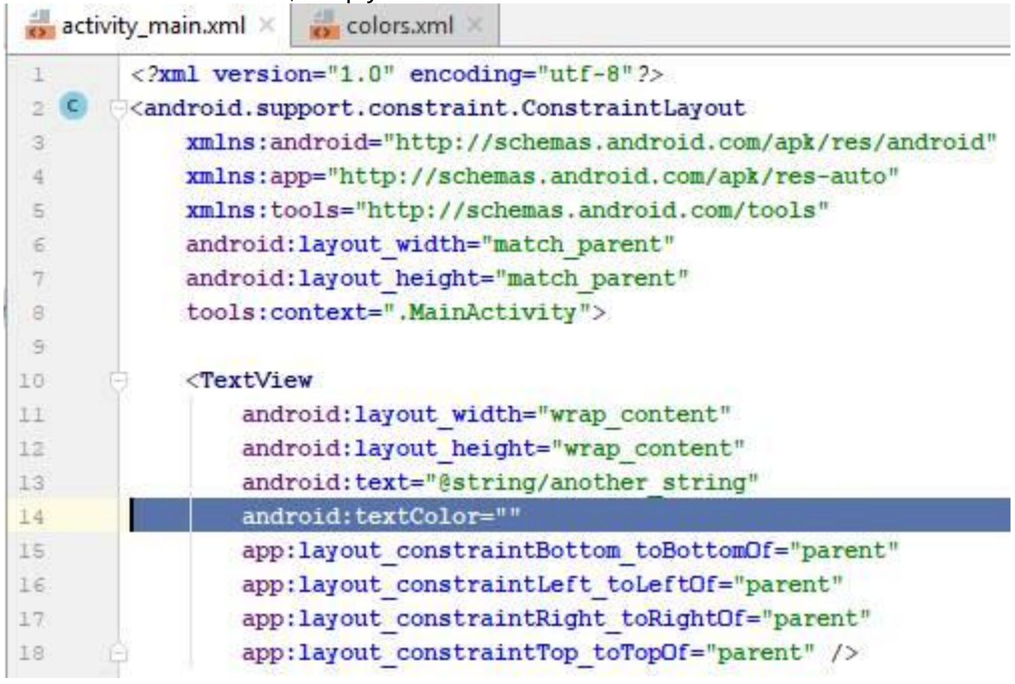
android:layout_margin
android:layout_marginBottom
android:layout_marginEnd
android:layout_marginHorizontal
android:layout_marginLeft
android:layout_marginRight
android:layout_marginStart
android:layout_marginTop
android:layout_marginVertical
android:accessibilityHeading
Press Ctrl+Space to view tags from other namespaces

- You can start typing the word `text` and it will narrow down the choices
- Note all the different attributes available for the `TextView` object



```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/another_string"
    android:text
```

- Scroll down and select `textColor` field indicating the color for the string
- Once this field is select, simply hit Enter to create the field in the XML file

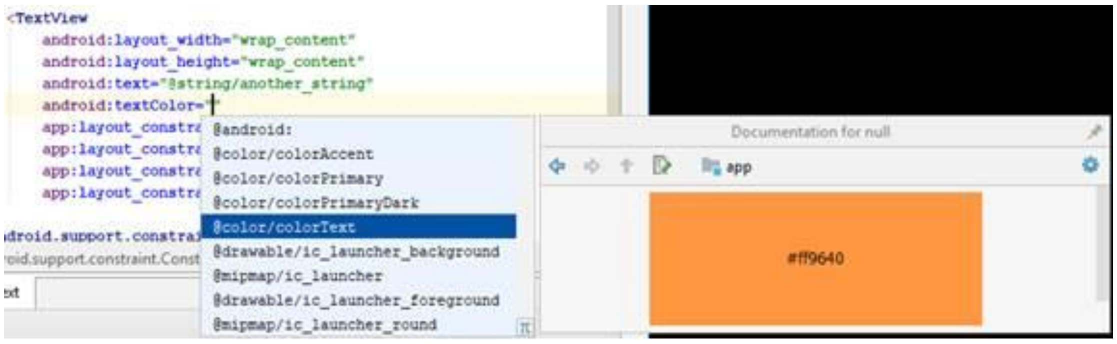


```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/another_string"
        android:textColor=""
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

- Inside the quotations of this new attribute, we can type Ctrl-Space again
- Android Studio offers a selection of resource type designations
- Since we are applying a text color, select the `@color/colorText` designation
- Android Studio offers selection from our available color resources

Zarqa Uni.
Internet Technology
Android Programming
Dr. Khaled Alqawasmi

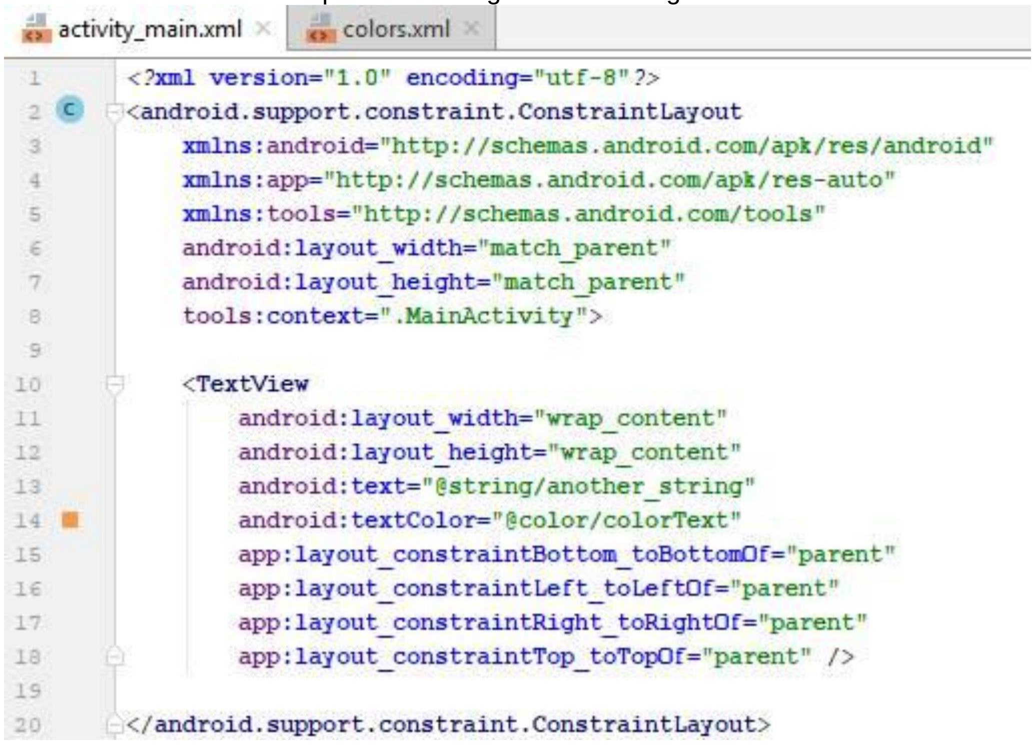


```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/another_string"
    android:textColor="
    app:layout_constra
    app:layout_constra
    app:layout_constra
    app:layout_constra

droid.support.constra
```

```
@android:
@color/colorAccent
@color/colorPrimary
@color/colorPrimaryDark
@color/colorText
@drawable/ic_launcher_background
@mipmap/ic_launcher
@drawable/ic_launcher_foreground
@mipmap/ic_launcher_round
```

- Select the new XML color resource variable we created, `textColor`
- Note the addition of the square indicating the color assigned

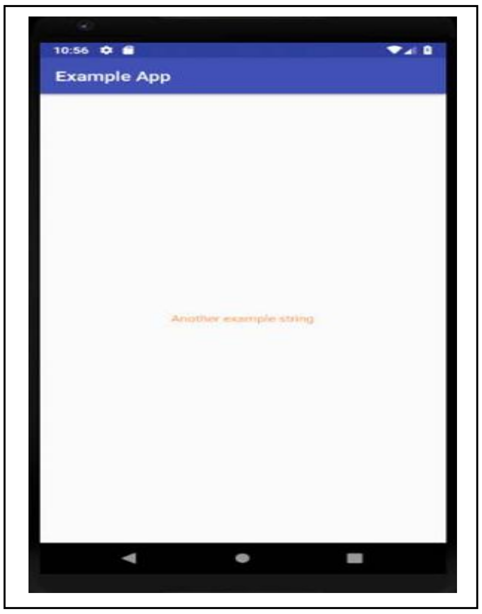


```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/another_string"
        android:textColor="@color/colorText"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

After saving the session and re-running our app, we see the updated change
- Note the color of our Activity string has changed to the resource color designated



10:56
Example App
Another example string