

Lecture #10

Advanced Mobile

Development

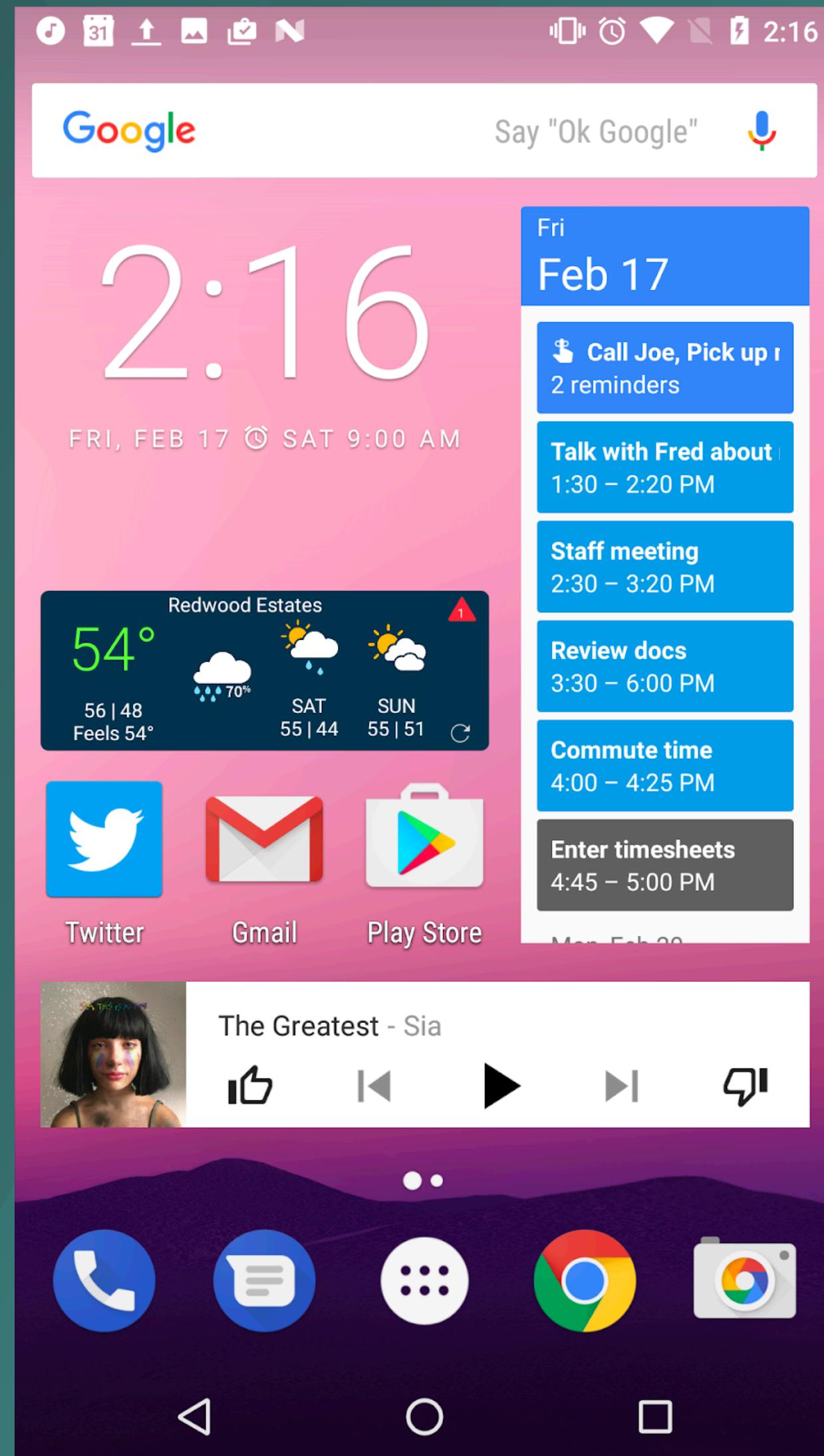
Mobile Applications
Fall 2023



La multi ani!

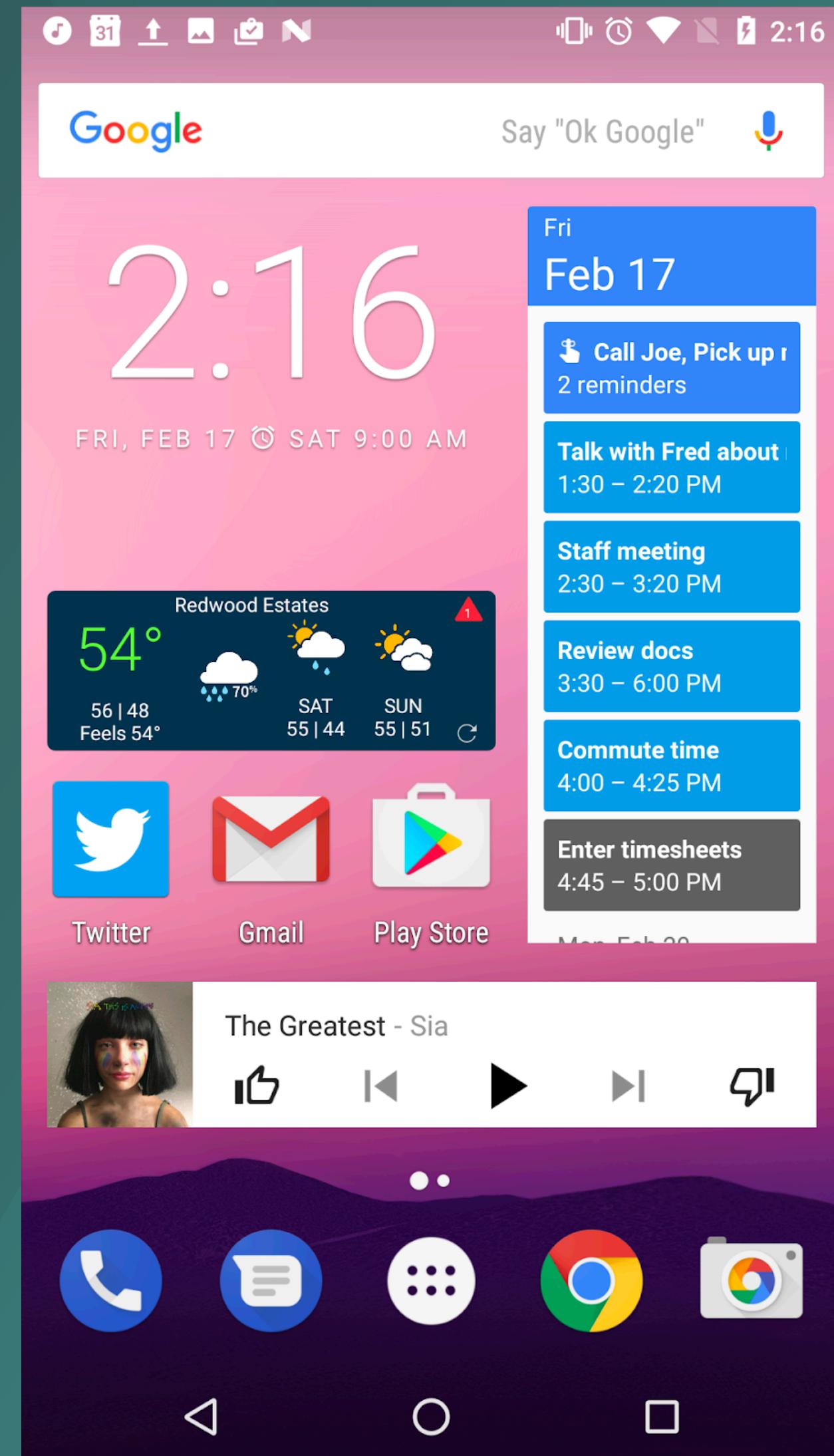
App Widgets

- A miniature app view.
- Runs on the home screen.
- Updated periodically.
- Display small amounts of information.
- Perform simple functions.



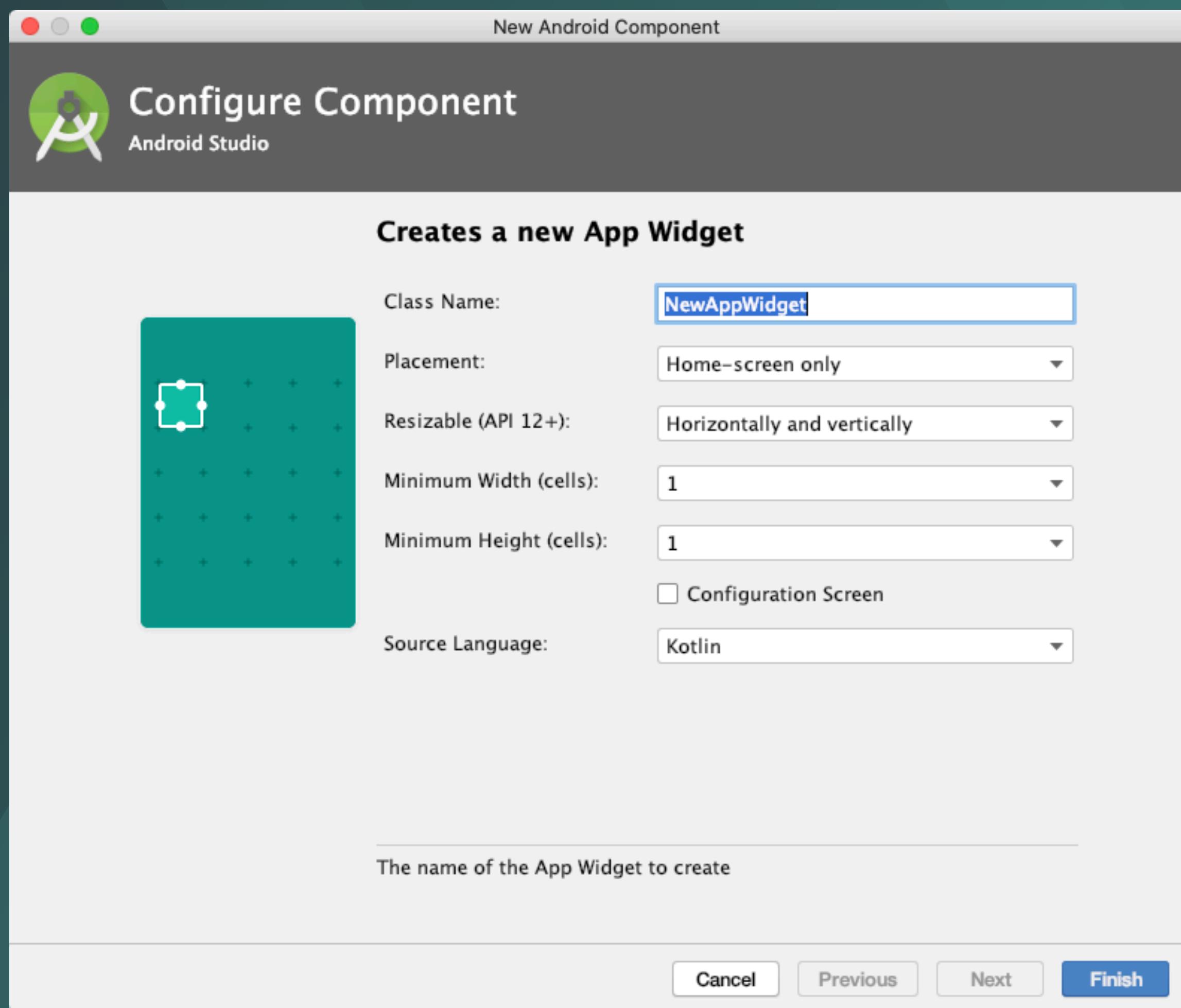
App Widgets

- Add-ons for an existing app.
- An app can have multiple widgets.
- Not available without an app.
- The default action is to start the app.

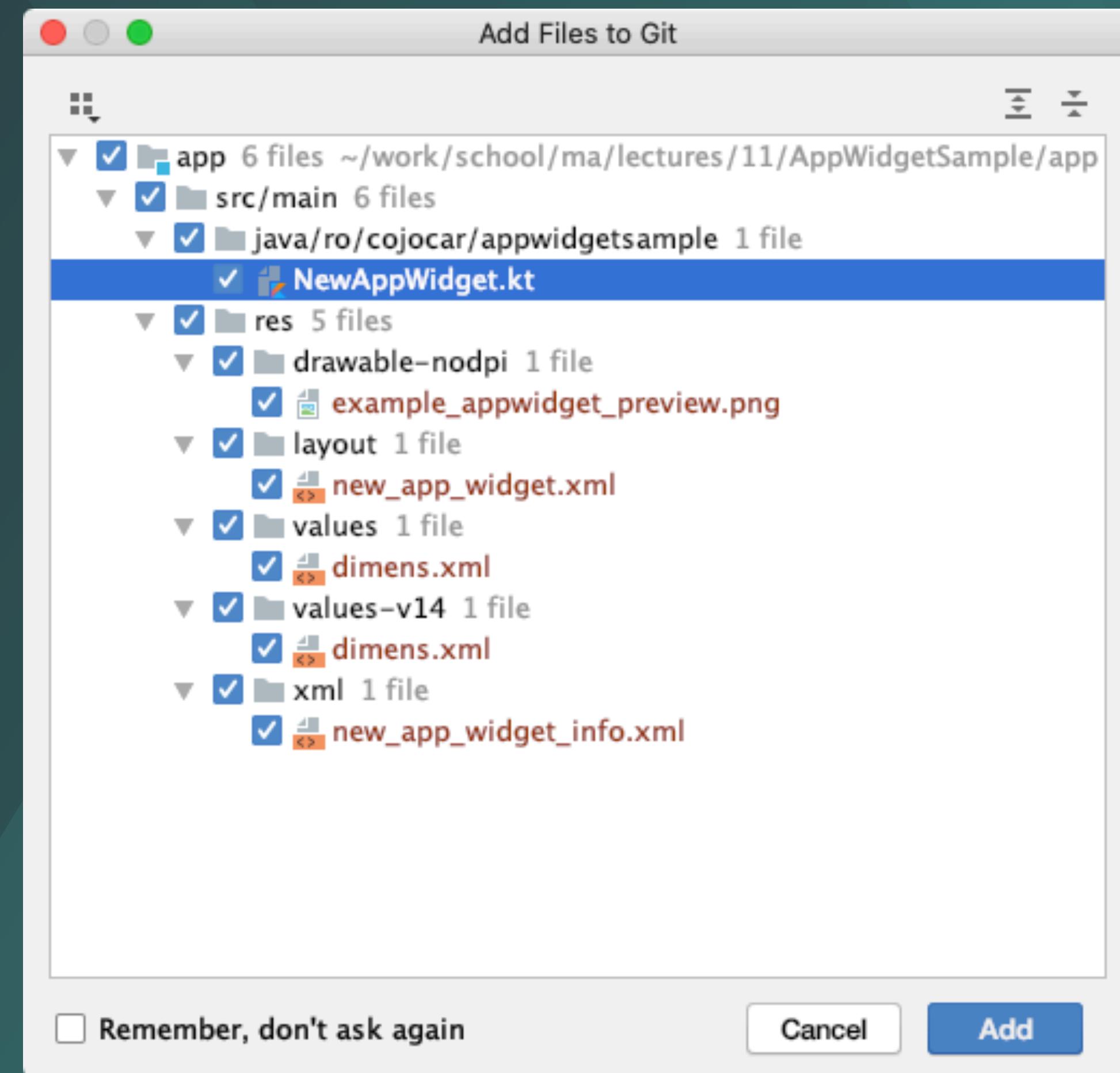


Set up the app widget project

- File > New > Widget > AppWidget.

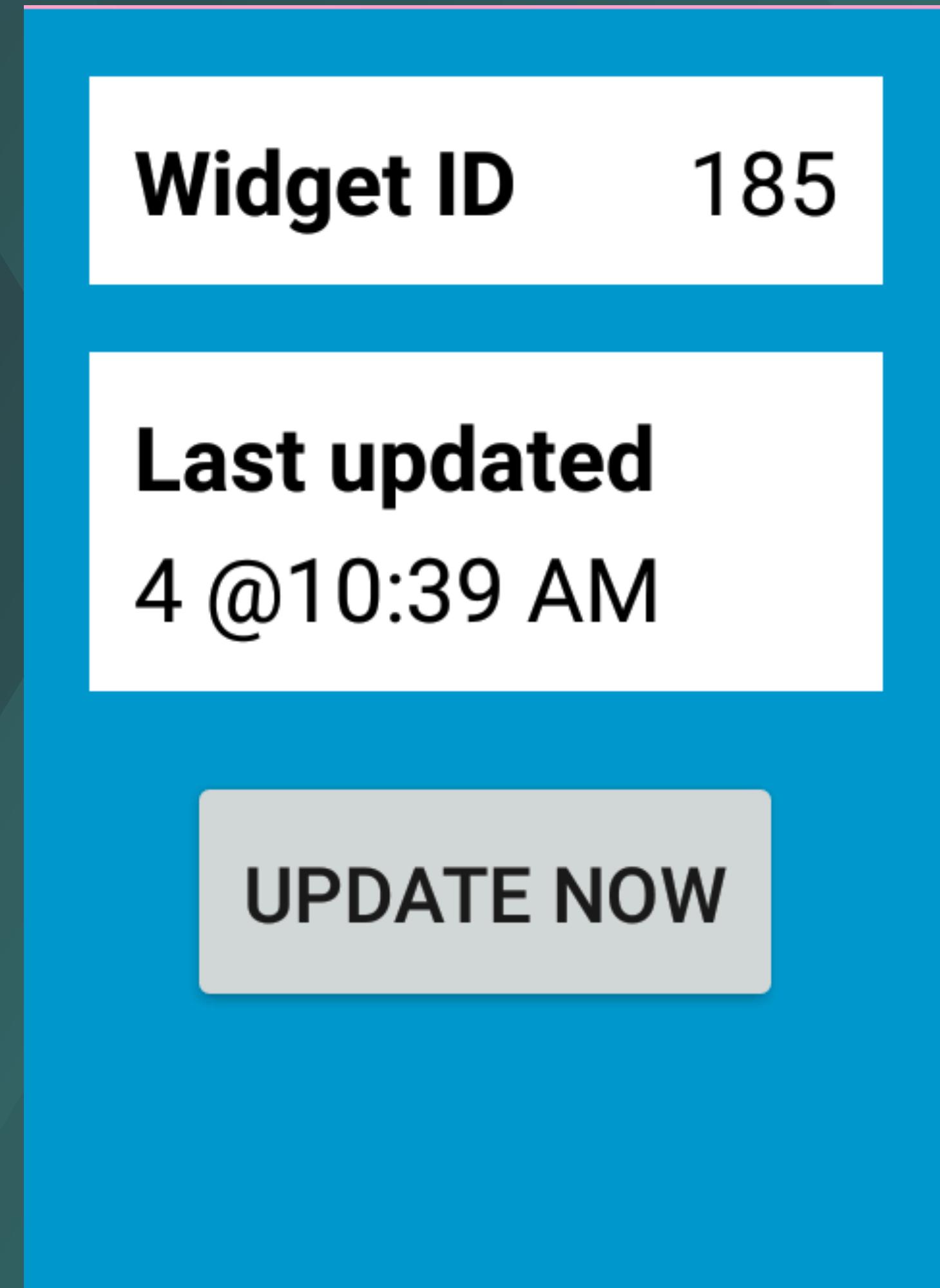


Generated Files

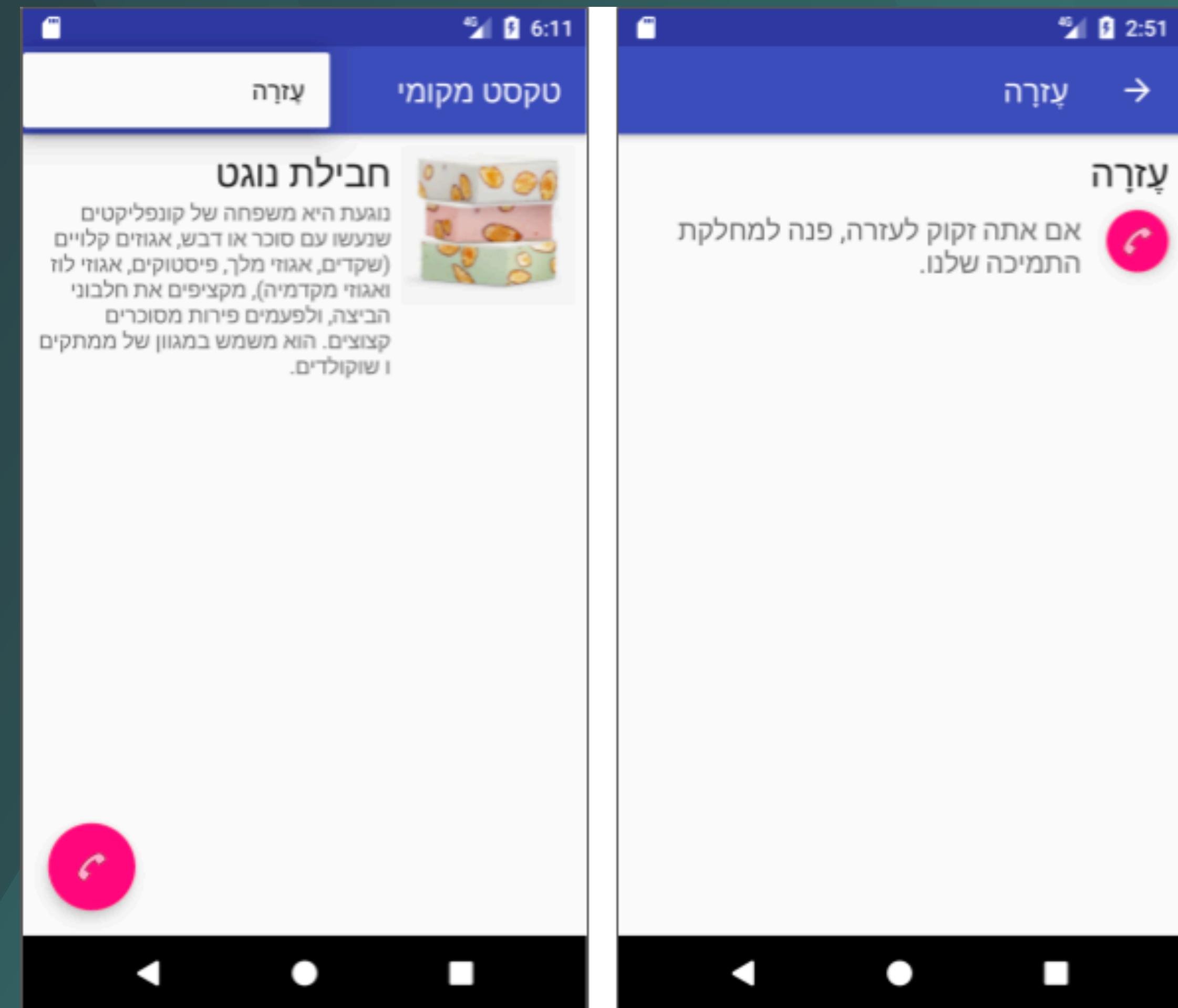


Customize the widget

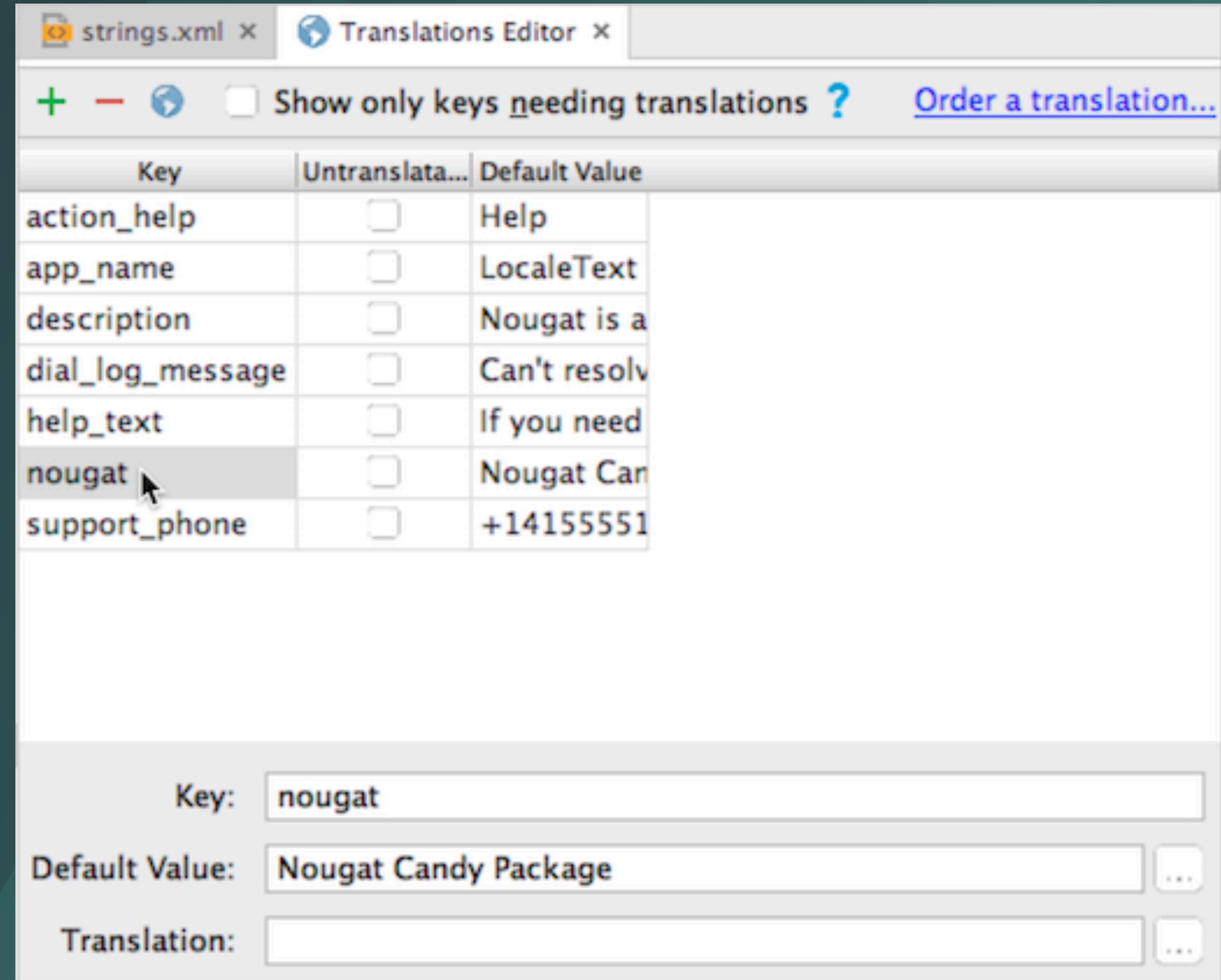
DEMO



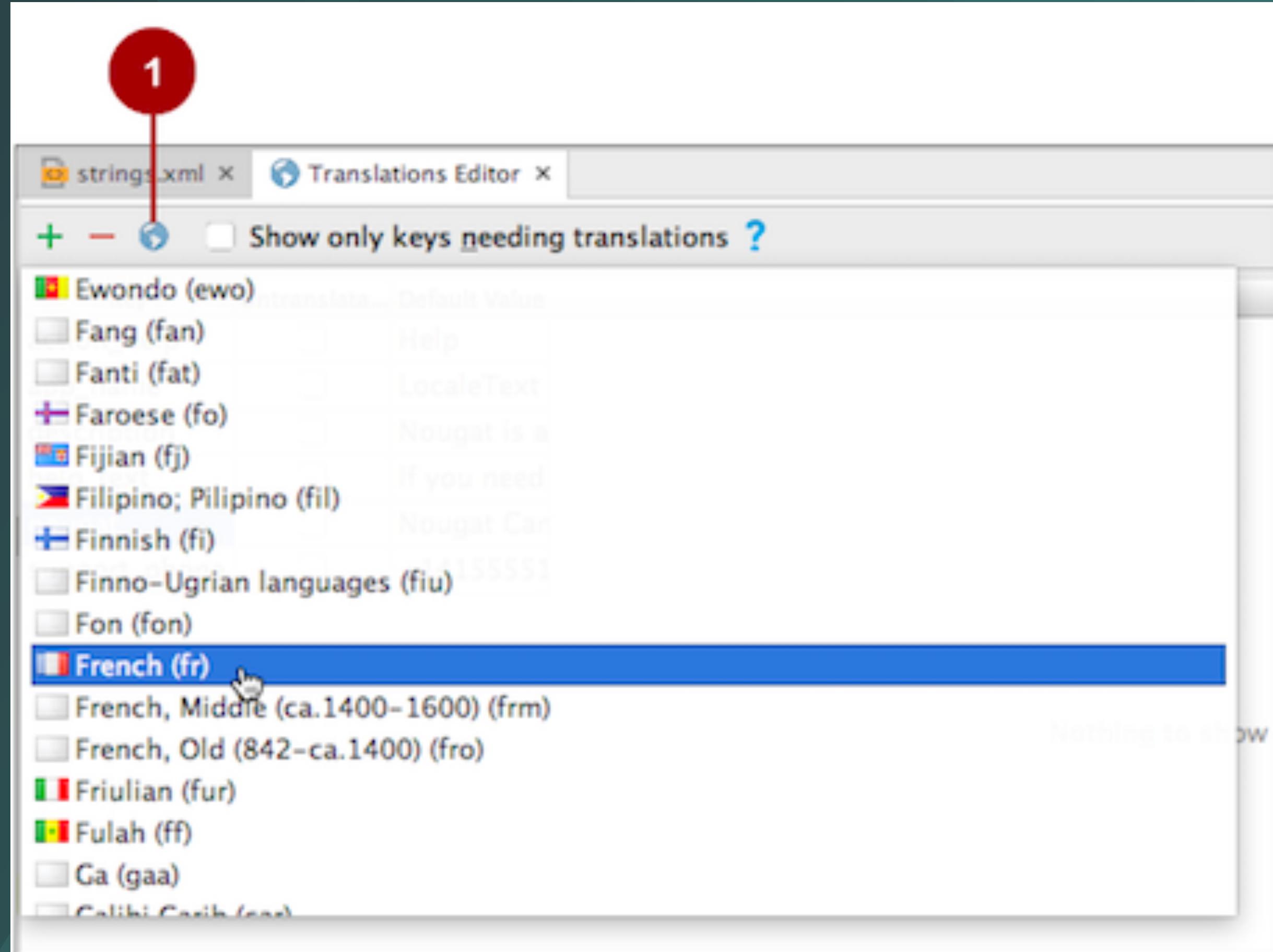
Language Support



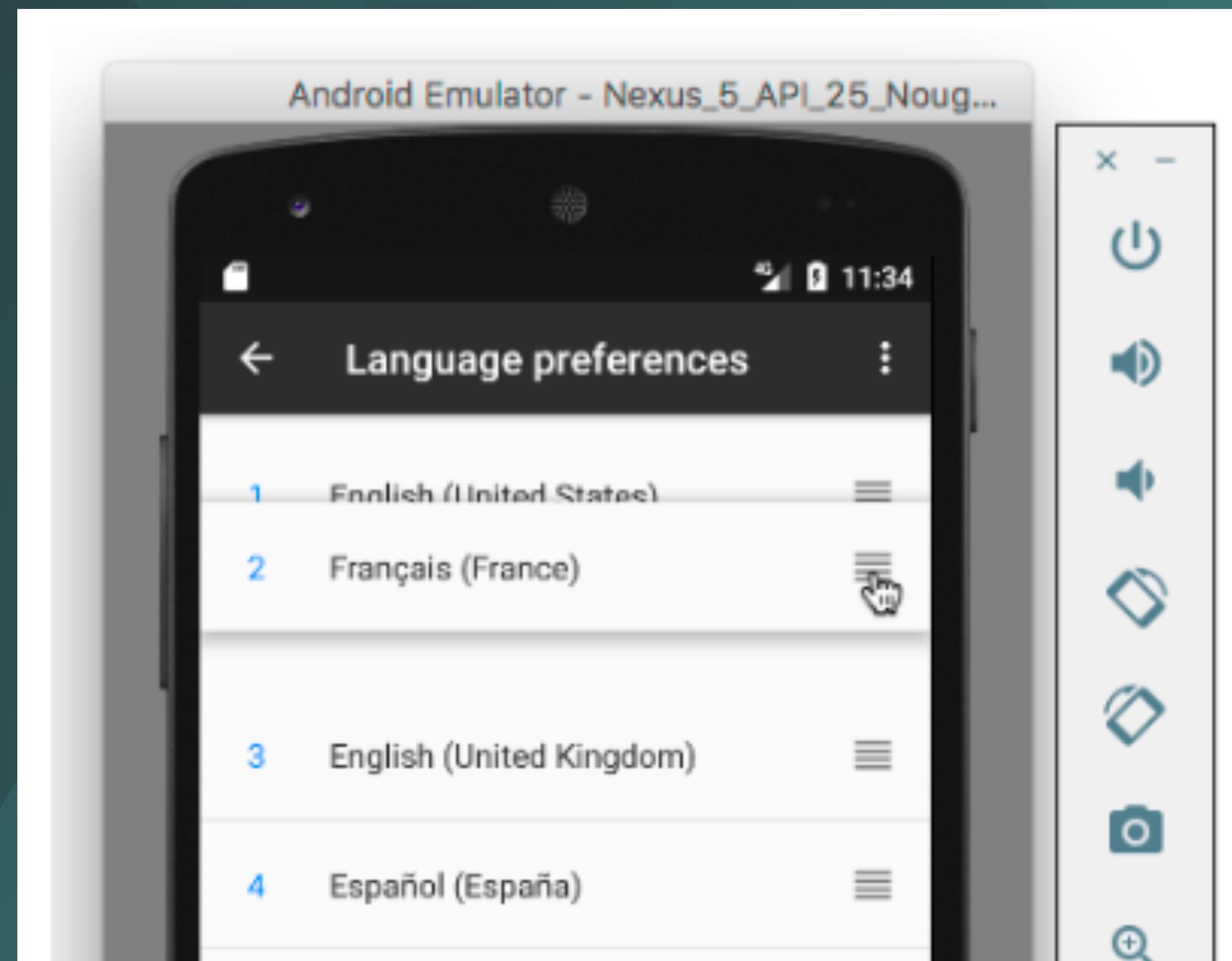
Add another language resource to the app



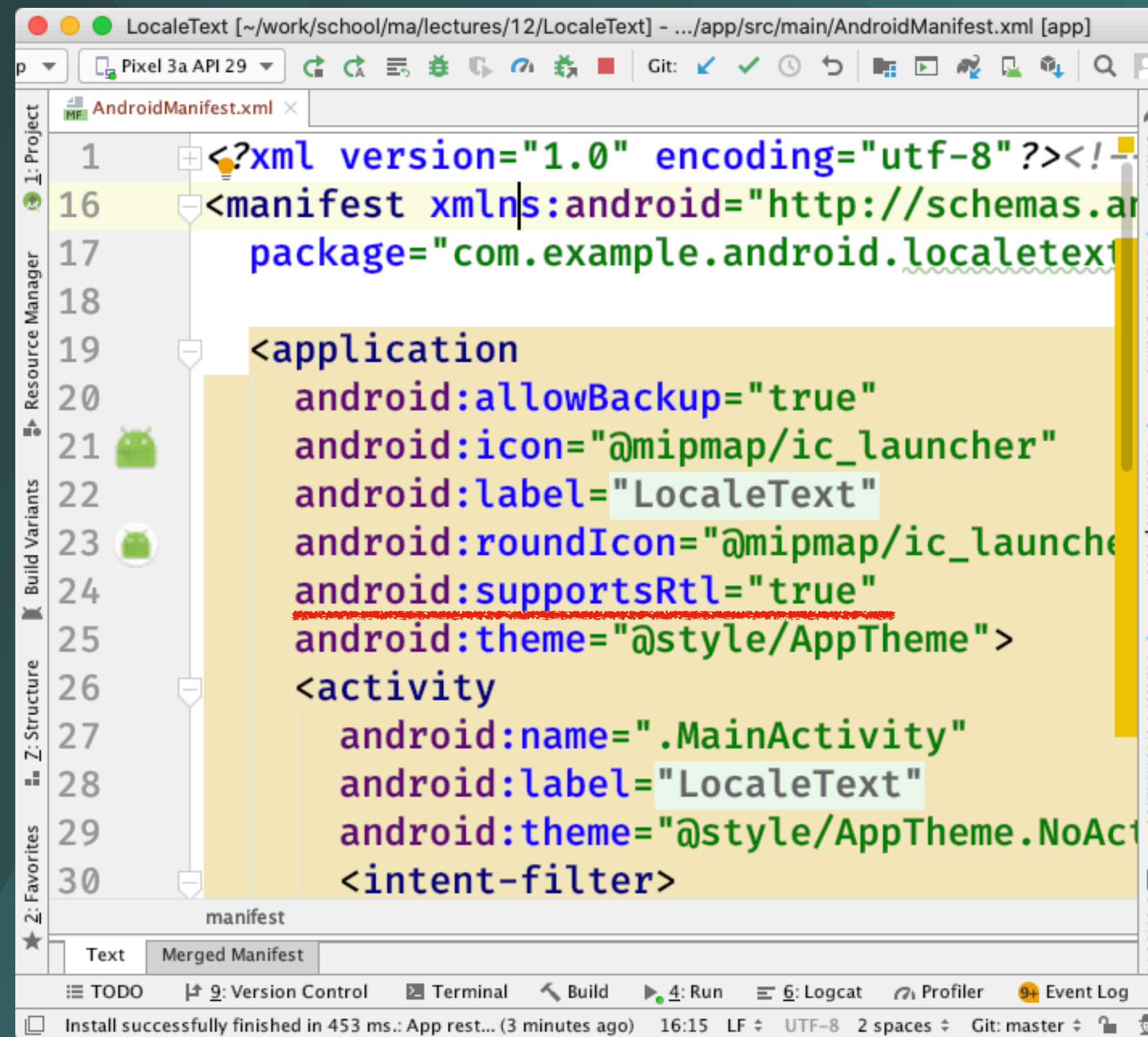
Add another language resource to the app



Run the app and switch languages



Add a right-to-left (RTL) language



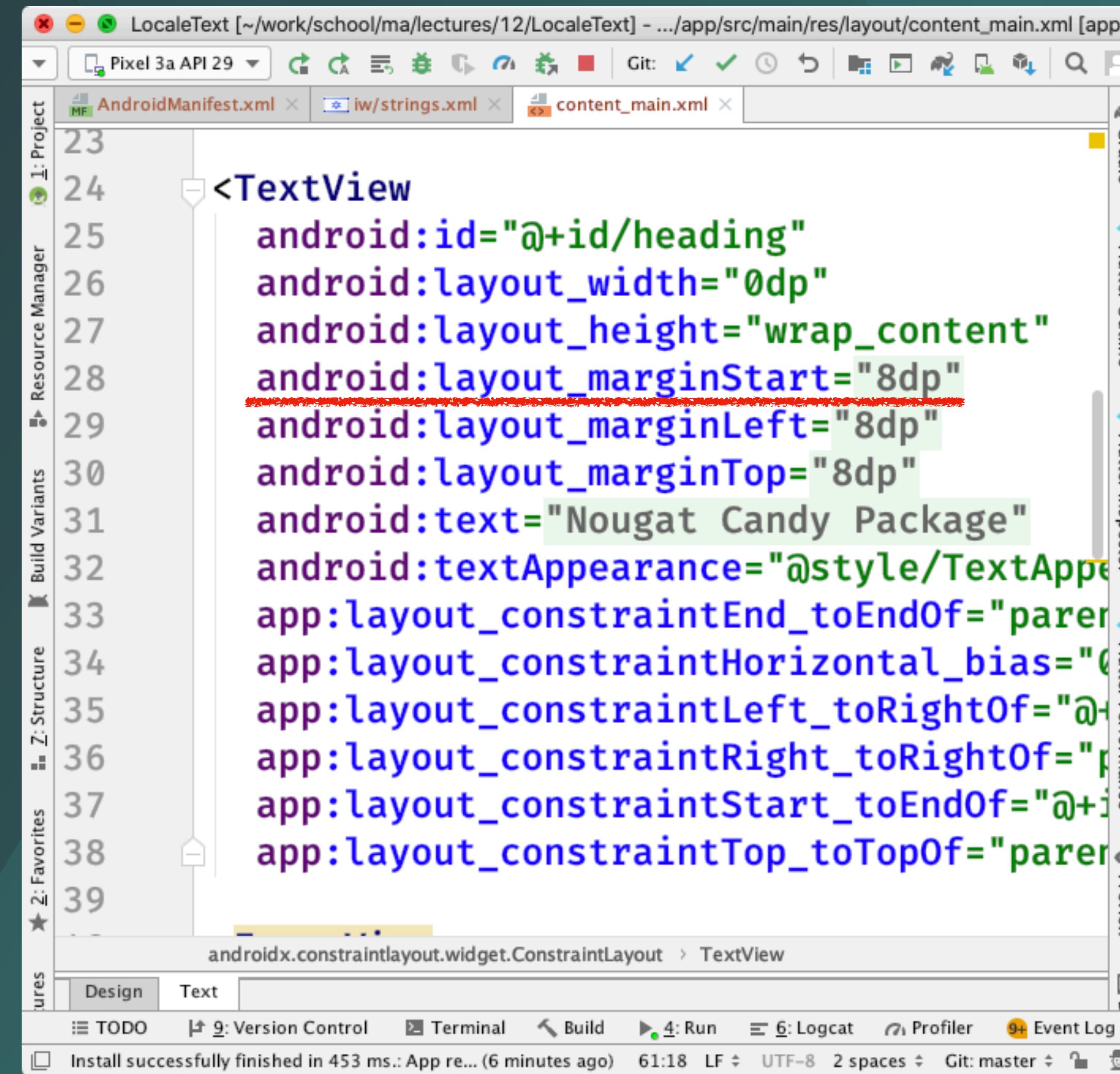
The screenshot shows the AndroidManifest.xml file in the Android Studio editor. The file defines an application with an activity. The key configuration for supporting RTL is the `android:supportsRtl="true"` attribute, which is highlighted with a red underline. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE manifest>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.localetext">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="LocaleText"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true" // This line is underlined in red
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="LocaleText"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
```

Add a right-to-left (RTL) language

DEMO



```
23
24     <TextView
25         android:id="@+id/heading"
26         android:layout_width="0dp"
27         android:layout_height="wrap_content"
28         android:layout_marginStart="8dp" // Red underline here
29         android:layout_marginLeft="8dp" // Red underline here
30         android:layout_marginTop="8dp"
31         android:text="Nougat Candy Package"
32         android:textAppearance="@style/TextAppe
33         app:layout_constraintEnd_toEndOf="parent"
34         app:layout_constraintHorizontal_bias="0.0"
35         app:layout_constraintLeft_toRightOf="@+id/
36         app:layout_constraintRight_toRightOf="P
37         app:layout_constraintStart_toEndOf="@+id/
38         app:layout_constraintTop_toTopOf="parent"
39     ...
```

The screenshot shows the Android Studio interface with the content_main.xml file open. The XML code defines a TextView with several layout attributes. The attributes android:layout_marginStart and android:layout_marginLeft are highlighted with a red underline, indicating they are incorrect for Right-to-Left (RTL) layout. The rest of the code is standard for a TextView.

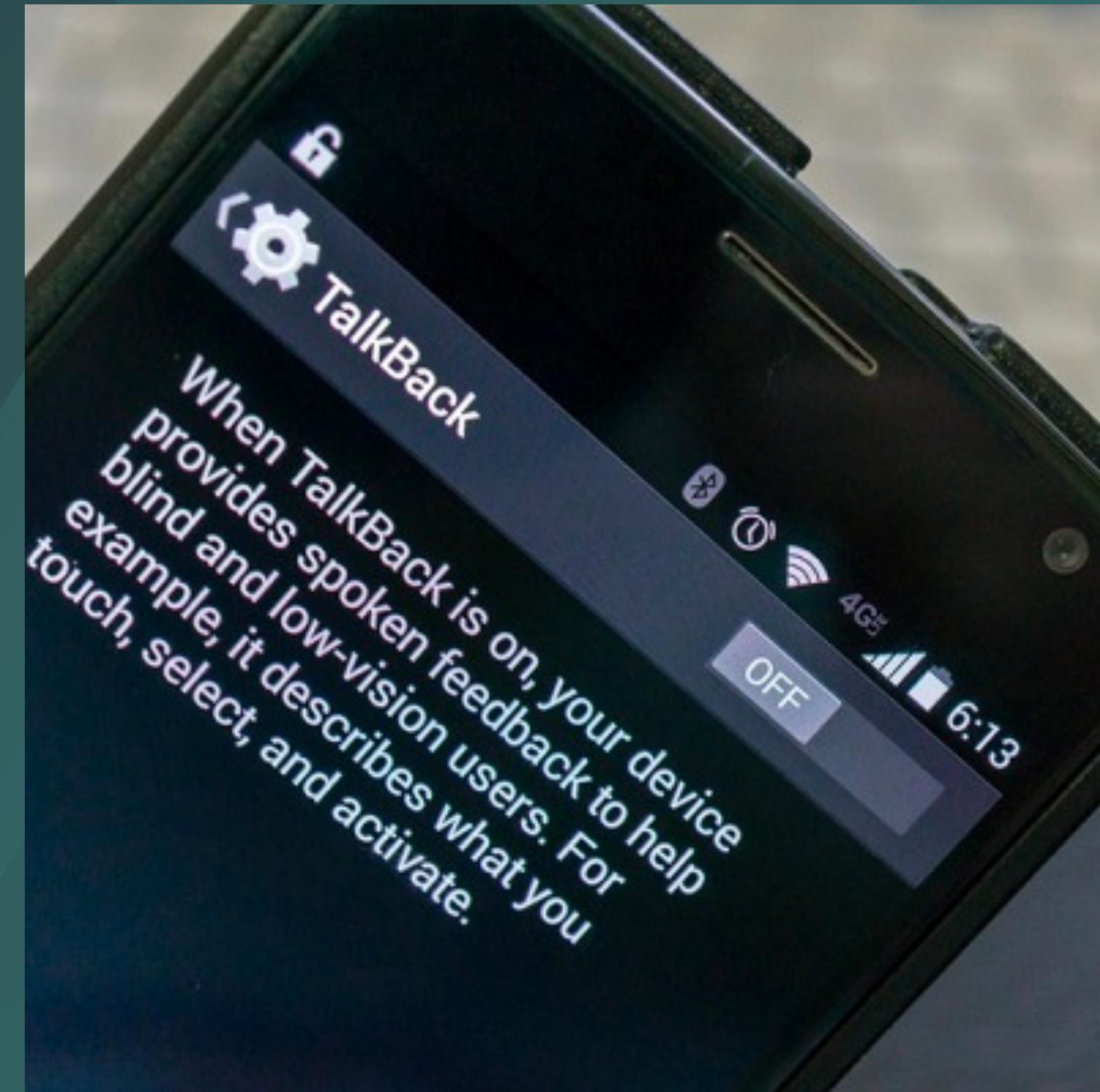
Accessibility

- Blindness
- Low vision.
- Color blindness.
- Deafness or hearing loss.
- Restricted motor skills.



TalkBack

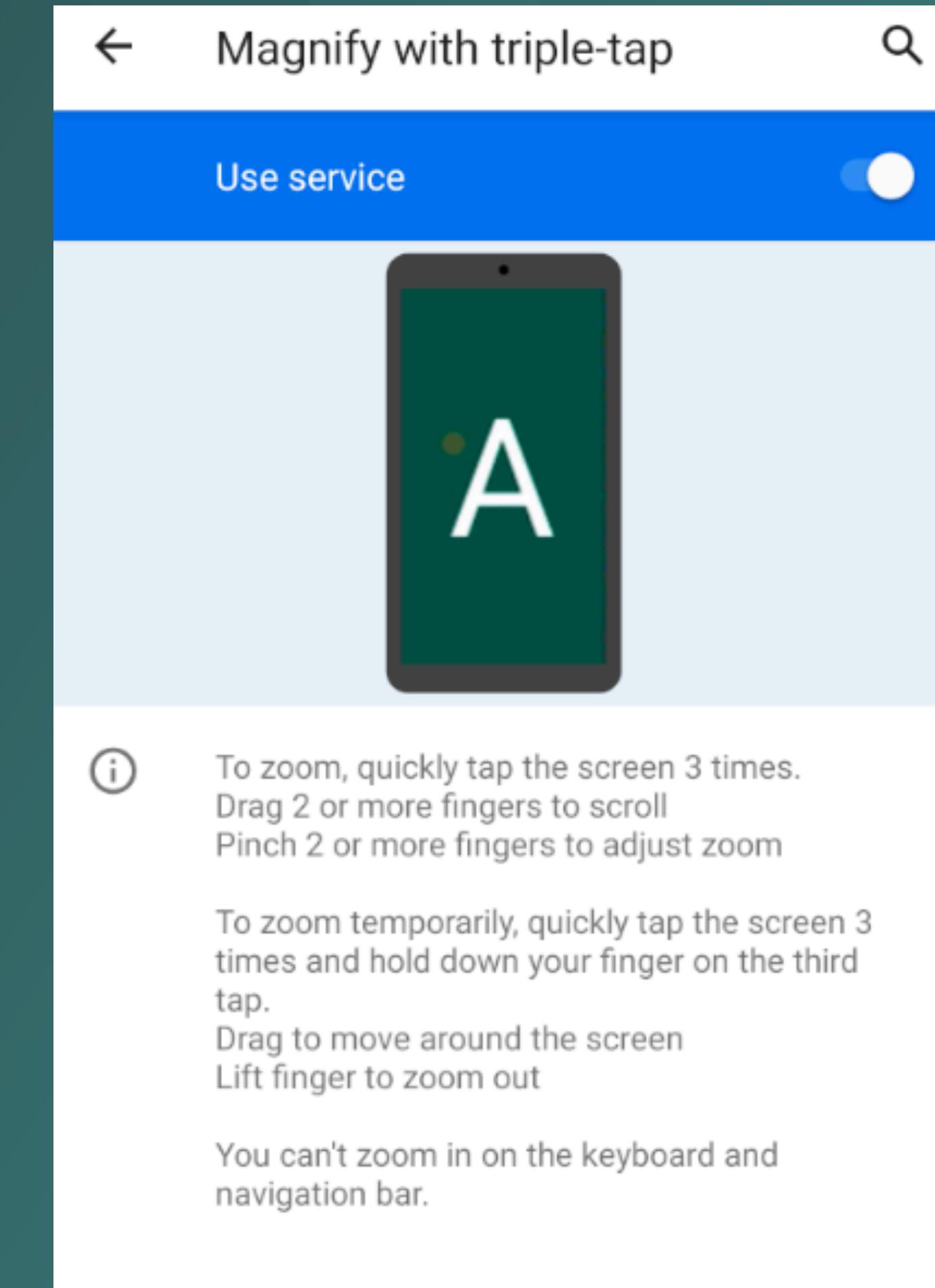
- Settings > Accessibility > TalkBack
- Settings > Accessibility > TalkBack > Settings > Launch TalkBack tutorial.



Font and Color

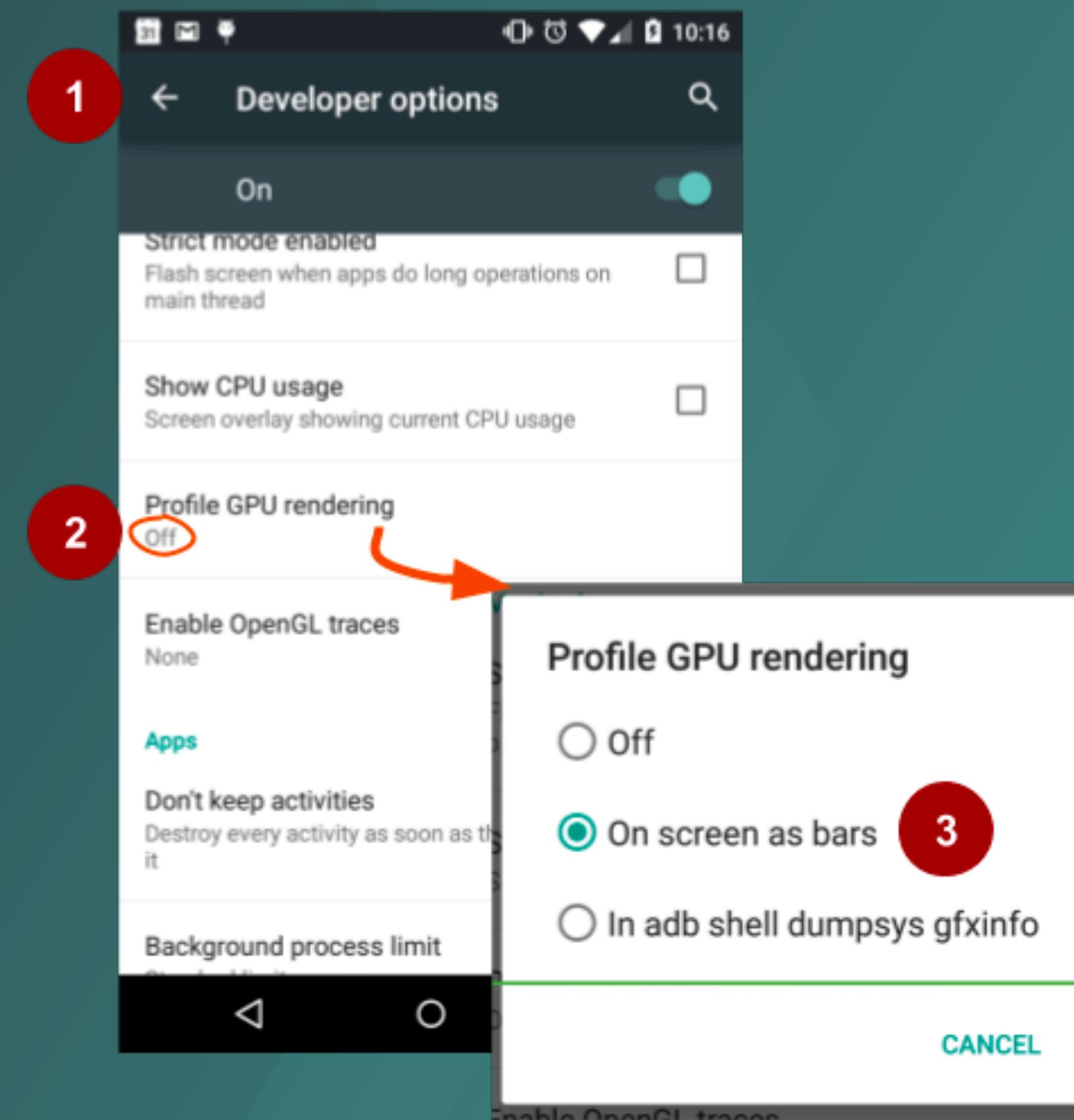
DEMO

- Settings > Accessibility > Magnification gesture.
- Settings > Accessibility > Font size.
- Settings > Accessibility > High contrast text.



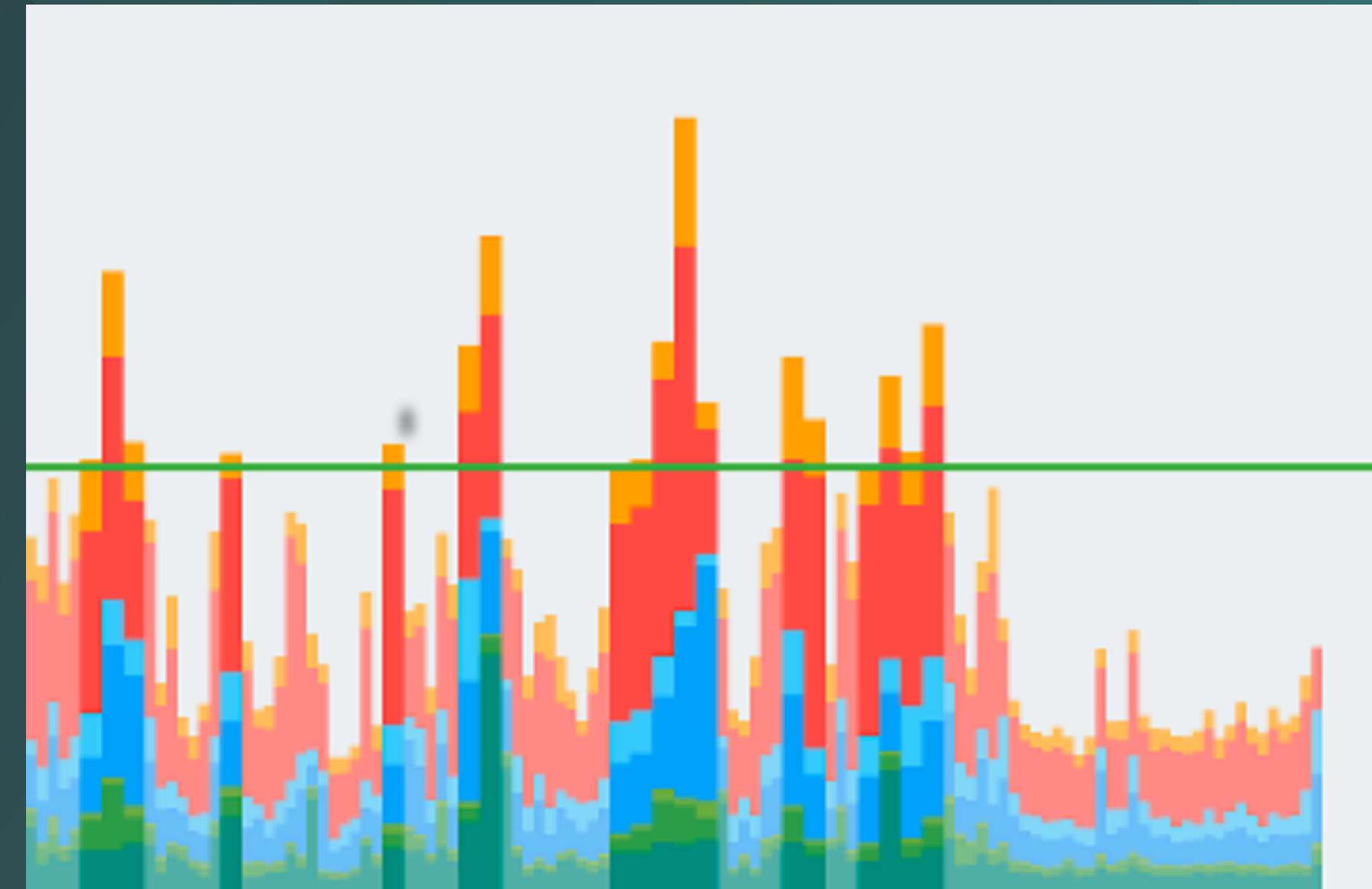
GPU Profiling

- Settings
 - Developer options
 - Monitoring
 - Profile GPU rendering



GPU Profiling

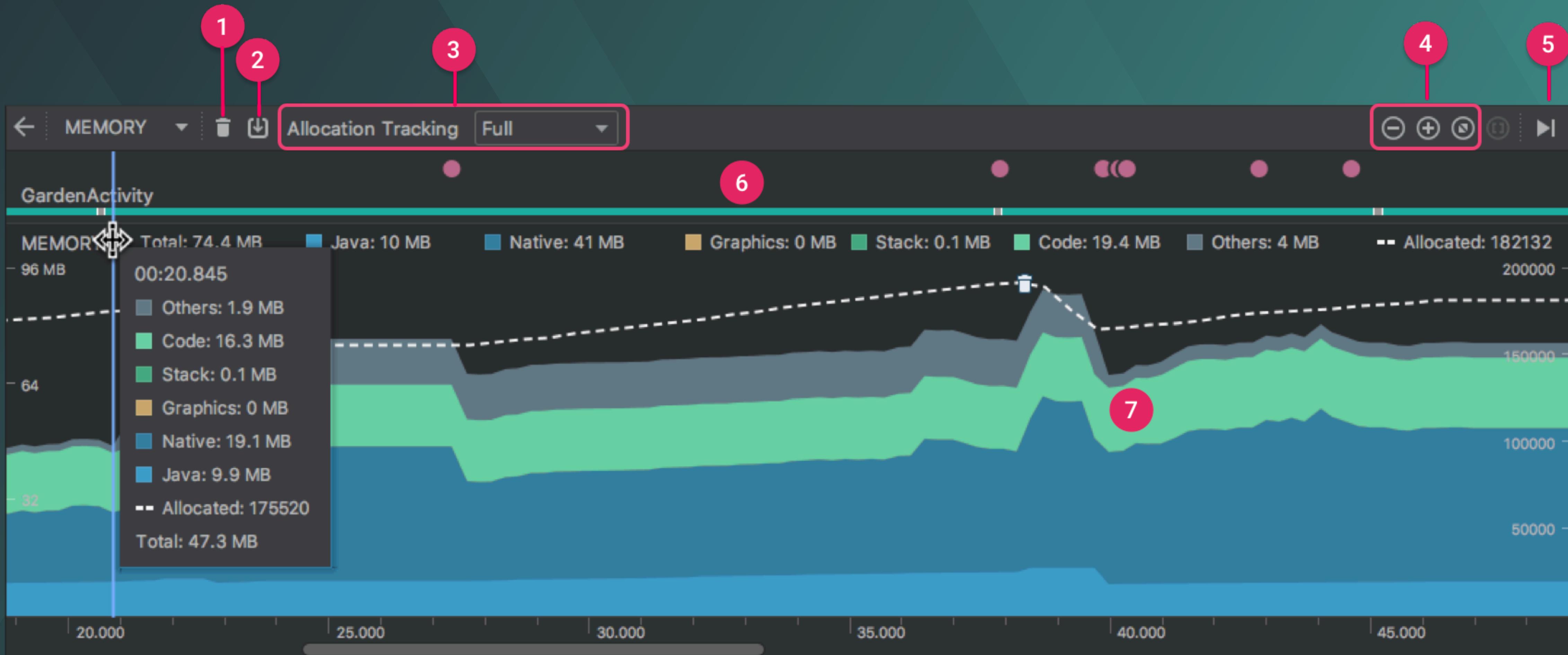
DEMO



Misc Input Anim. Measure Draw Upload Issue Swap

Memory Profiling

DEMO



ViewBinding

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

    <TextView
        android:id="@+id/mainTitle"
        tools:text="Main Title" />

    <TextView
        android:id="@+id/subTitle"
        tools:text="Main Subtitle" />
</RelativeLayout>
```

ViewBinding

```
public class MainActivity extends AppCompatActivity {  
    private TextView txtViewMainTitle;  
    private TextView txtViewSubTitle;  
  
    @Override  
    protected void onCreate(@Nullable Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        txtViewMainTitle = findViewById(R.id.mainTitle);  
        txtViewSubTitle = findViewById(R.id.subTitle);  
  
        txtViewMainTitle.setText("This is my main title");  
        txtViewSubTitle.setText("This is my subTitle");  
    }  
}
```

ViewBinding

Android
Open Source Project CHANGES ▾ DOCUMENTATION ▾ BROWSE ▾

Merged as [637b173](#) | [882241](#): Sample updates: Fragment state, synth accessors

Updated	Jan 30, 2019
Owner	Jakub Gielzak
Assignee	
Reviewers	Treehugger Robot Florina Muntenescu Jelle Fresen
CC	Nikita Frukt
Repo	platform/frameworks/support
Branch	androidx-master-dev
Parent	2a9664f
Topic	No topic
Hashtags	

Sample updates: Fragment state, synth accessors

1) Moved click count to Fragment state
This verifies / highlights FragmentStateAdapter's ability to correctly handle Fragment state.

2) Replaced kotlinx synthetic with findViewById
kotlinx.android.synthetic is no longer a recommended practice. Removing in favour of explicit findViewById.

Bug: [122659289](#)
Test: manual

Change-Id: [Ic472f90e28f7133822edcf53f44b83dc333f768e](#)

Code-Review +1 Florina Muntenescu
+2 Jelle Fresen

ViewBinding

The Argument Over Kotlin Synthetics

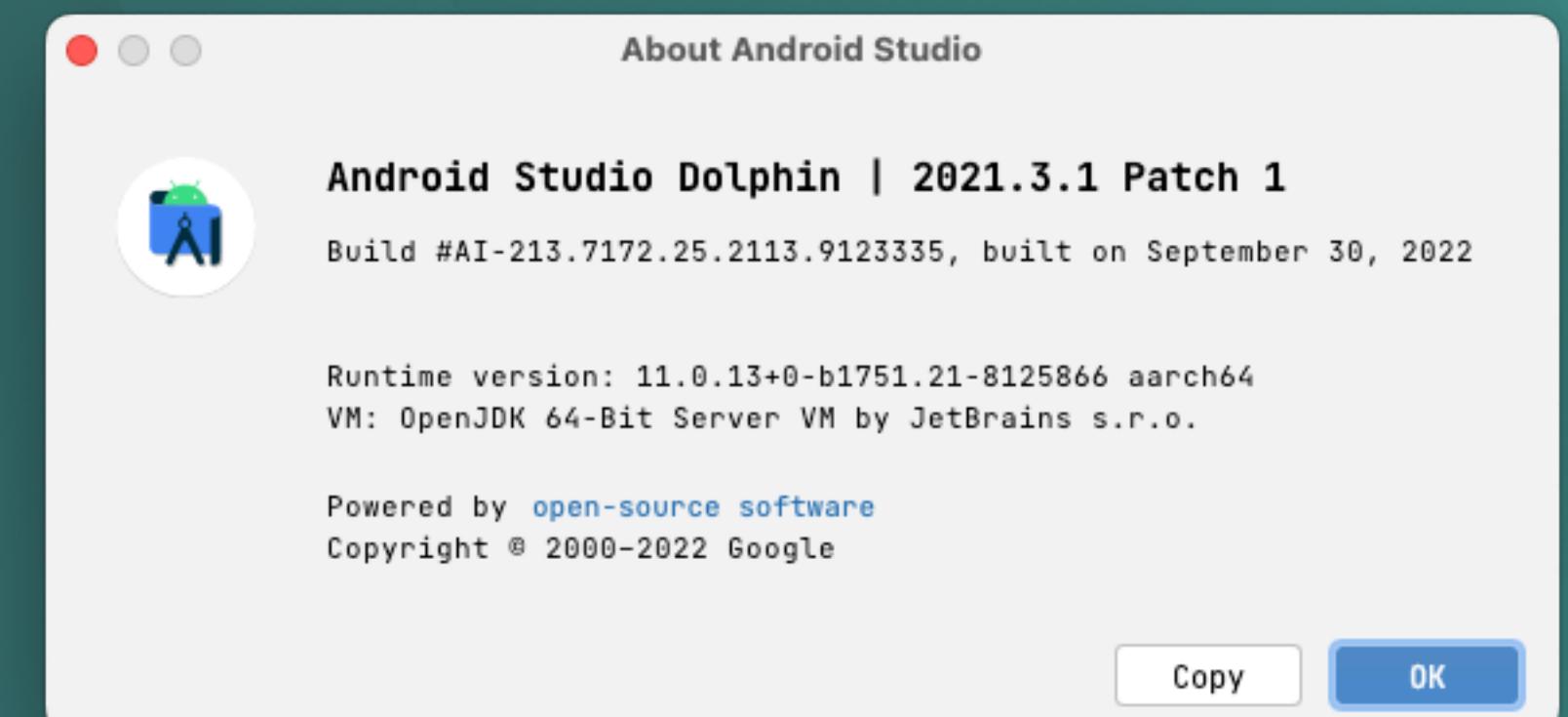
- They Are Kotlin Only.
- They Don't Expose Nullability.
- The Code Generated Is Not Guaranteed To Be Performant.
- Everything Exists In A Global Namespace.
- Typing Isn't Guaranteed.

ViewBinding

Android Studio 3.6 Canary 11+.

app/build.gradle:

```
android {  
    ...  
    viewBinding {  
        enabled = true  
    }  
}
```



ViewBinding

build.gradle:

```
buildscript {  
    ext.kotlin_version = '1.7.20'  
    repositories {  
        google()  
        mavenCentral()  
    }  
    dependencies {  
        classpath "com.android.tools.build:gradle:7.3.1"  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
    }  
}
```

ViewBinding

DEMO

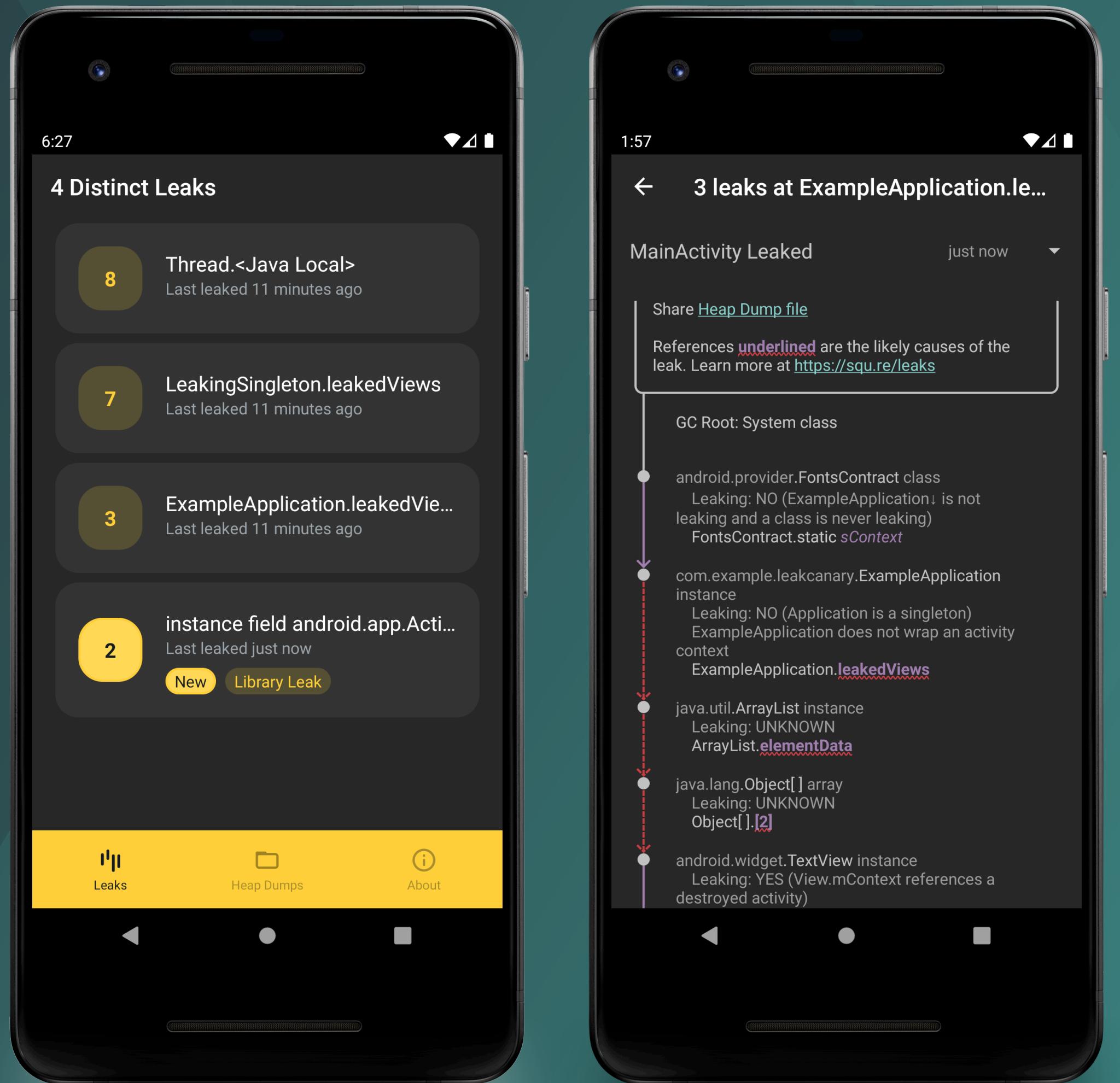
- The auto-generated .class increases app size.
- Harder to debug.
- Null safety.
- Type safety.
- Speed.



Memory Leaks

“A small leak will sink a great ship.” - Benjamin Franklin

Memory Leaks



<https://square.github.io/leakcanary>

Memory Leaks

```
dependencies {  
    // debugImplementation because LeakCanary should only run in debug builds.  
    debugImplementation 'com.squareup.leakcanary:leakcanary-android:2.10'  
}
```

Memory Leaks

DEMO

```
dependencies {  
    // debugImplementation because LeakCanary should only run in debug builds.  
    debugImplementation 'com.squareup.leakcanary:leakcanary-android:2.10'  
}
```

LeakCanary automatically detects leaks of the following objects:

- Destroyed Activity instances.
- Destroyed Fragment instances.
- Destroyed fragment View instances.
- Cleared ViewModel instances.
- Destroyed Service instance.

Lecture outcomes

- Identify app widgets, and understand the key parts of an app widget.
- Implement app widget actions when an element of an app widget is tapped.
- Add support for different languages.
- Test your app for accessibility in a variety of ways.
- Use the Profile GPU Rendering tool to visualize Android drawing the screen.
- Use Memory Profiler to collect data about your app.

