Android

Android Studio

Kotlin Language – xml for design

Eng. Ibrahim AlDekhail BCT - TVTC

Introduction

1. What is Kotlin?

Kotlin is a modern programming language developed by JetBrains. It is statically typed and runs on the Java Virtual Machine (JVM). The Kotlin language is designed to be interoperable with Java, so you can use Kotlin code alongside Java code seamlessly. Aside from Android app development, it is also used for server-side and web development.

2. How is Kotlin different from Java?

Kotlin and Java both programming languages that run on the Java Virtual Machine (JVM).

But Kotlin is designed to be more concise and expressive, reducing unnecessary code.

One notable feature is its built-in null safety, helping to avoid common programming errors related to null values.

Kotlin also introduces modern language features like extension functions and coroutines, offering developers more flexibility and improved productivity compared to Java.

Introduction

1. Java vs Kotlin: Difference Between Kotlin and Java

Java and Kotlin are two widely known programming languages used for developing Android applications. Both these languages are quite similar; however, while drawing the comparison between Java vs Kotlin, there are some of the key differences that you should consider. Let's have a look at them.

Comparison Factors	Java	Kotlin
Popularity	Widely Used	Steadily Growing
Ease of Use	More Concise	Mature Development Environment
Scalability	Yes	Yes
Community Support	Well-established Community	Still Expanding

Introduction

Java is Better For:

- Large applications that require multiple features and functionalities and need to function across all platforms, such as Windows, iOS, Android, or Linux.
- Java also has mature libraries that support this type of application development well and provide the integration of features with the latest updates.

Kotlin is Better For:

- Android app development with Kotlin is better suited for apps where performance matters a lot, such as those that are required to run smoothly on older Android phones or the ones that are used for photo editing.
- In Java vs Kotlin, Kotlin has a more streamlined and efficient design than Java, so it will perform better in these types of situations, particularly when scaling is concerned
- Kotlin is able to perform these functions while Java cannot because of its use of bytecode which can only compile code for one specific platform at once.

Key Features of Java

- Platform Independence
- Object-Oriented Programming
- Security
- Standard Library
- Large and Active Community

Key Features of Kotlin

- Conciseness and Readability
- Null Safety
- Interoperability with Java
- Functional Programming Support
- Coroutines

Q5: What is the difference between `var` and `val` in Kotlin?

Answer

- **var** is like `general` variable and it's known as a *mutable* variable in kotlin and can be assigned multiple times.
- **val** is like `Final` variable and it's known as *immutable* in Kotlin and can be initialized only single time.

```
+----------------+----------------------------+---------------------------+
|                |            val             |           var             |
+----------------+----------------------------+---------------------------+
| Reference type | Immutable(once initialized | Mutable(can able to change|
|                | can't be reassigned)       | value)                    |
+----------------+----------------------------+---------------------------+
| Example        | val n = 20                 | var n = 20                |
+----------------+----------------------------+---------------------------+
| In Java        | final int n = 20;          | int n = 20;               |
+----------------+----------------------------+---------------------------+
```

Q6: Where should I use `var` and where `val` ?

Answer

Use **var** where value is changing *frequently*. For example while getting location of android device:

```
var integerVariable : Int? = null
```

Use **val** where there is *no change* in value in whole class. For example you want set textview or button's text programmatically.

```
val stringVariables : String = "Button's Constant or final Text"
```

Q17: What is `lateinit` in Kotlin and when would you use it?

Answer

lateinit means *late initialization*. If you do not want to initialize a variable in the constructor instead you want to initialize it later on and if you can guarantee the initialization before using it, then declare that variable with lateinit keyword. It will not allocate memory until initialized. You cannot use lateinit for primitive type properties like Int, Long etc.

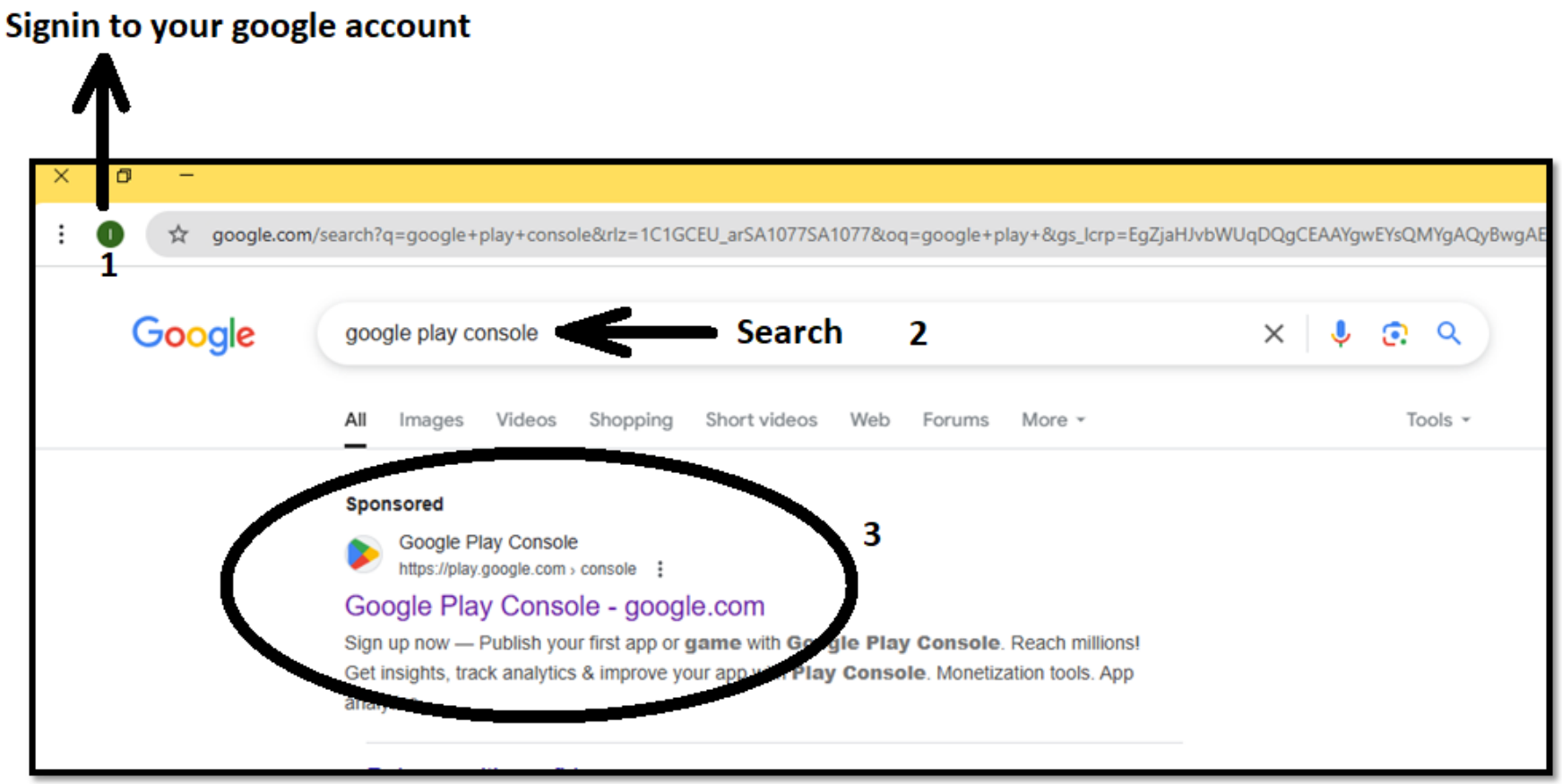
```
lateinit var test: String

fun doSomething() {
    test = "Some value"
    println("Length of string is "+test.length)
    test = "change value"
}
```

There are a handful of use cases where this is extremely helpful, for example:

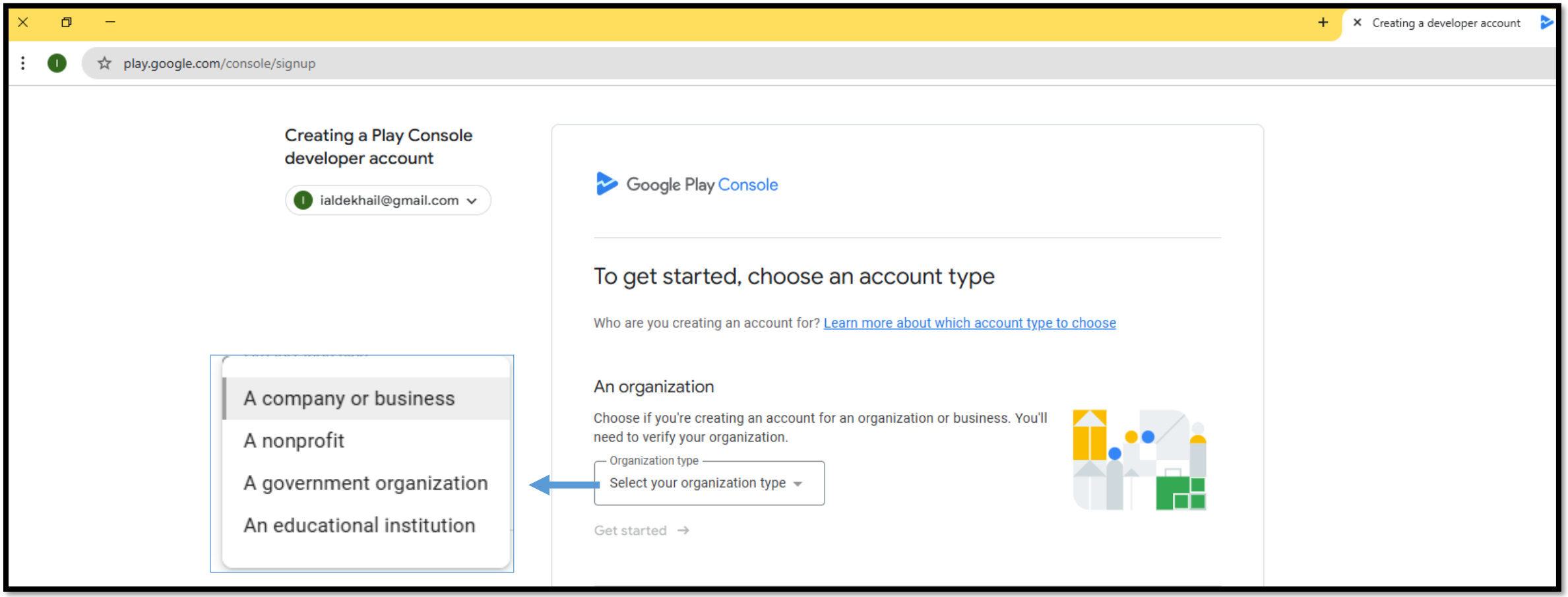
- Android: variables that get initialized in lifecycle methods;
- Using Dagger for DI: injected class variables are initialized outside and independently from the constructor;
- Setup for unit tests: test environment variables are initialized in a `@Before` - annotated method;
- Spring Boot annotations (eg. `@Autowired`).

Google Play Console

Create new Google Play Console Account

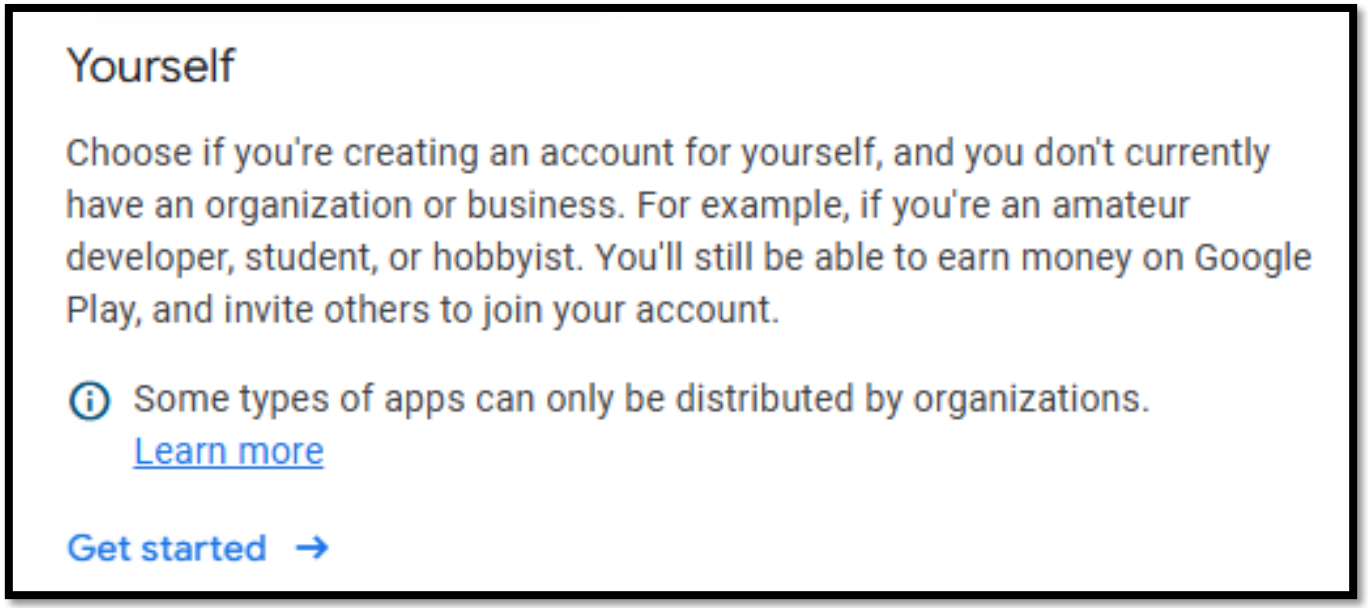


Create new Google Play Console Account



https://play.google.com/console/signup

Create new Google Play Console Account

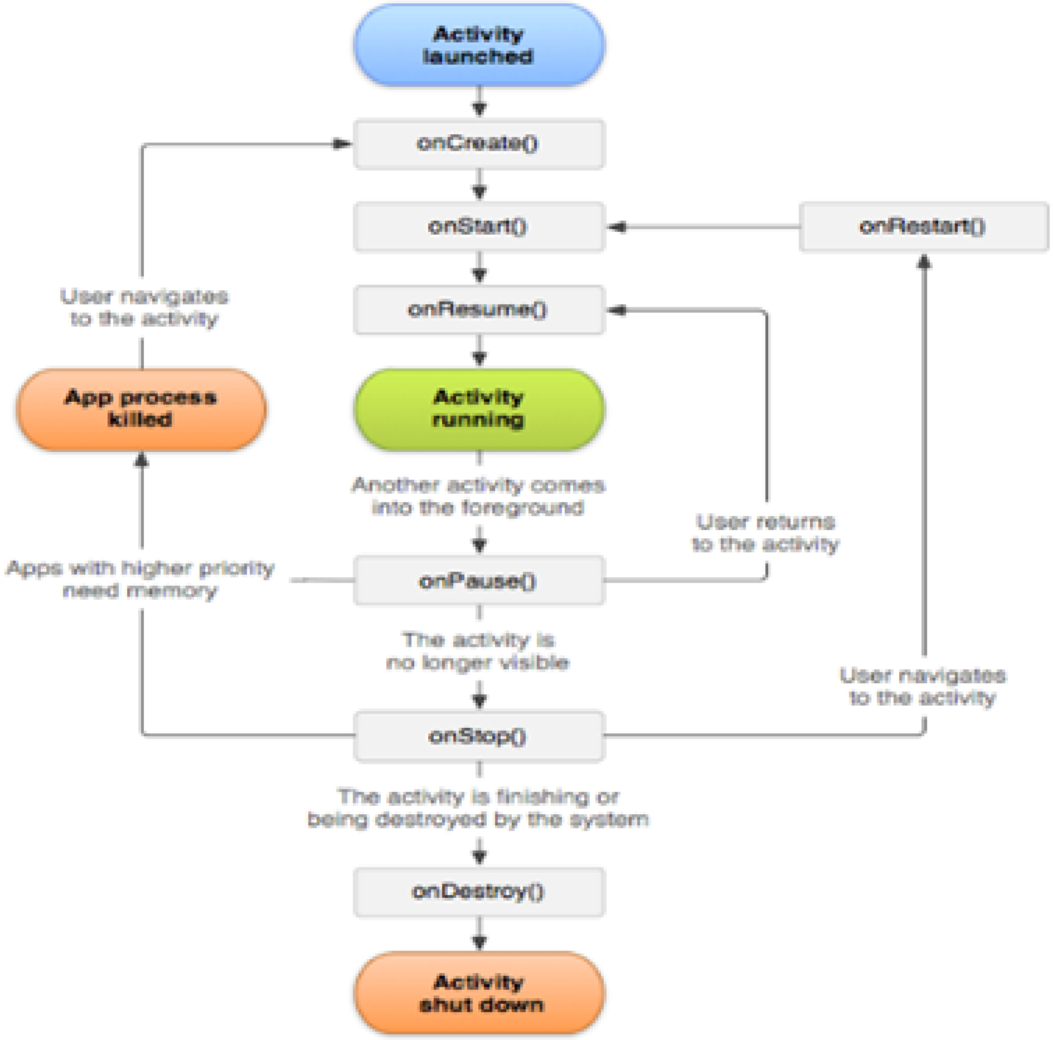


Android activity Lifecycle methods

Android Activity Lifecycle methods

Let's see the 7 lifecycle methods of android activity.

Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.



الدالة	متى تستدعاء
onCreate	تستدعاء عند أول انشاء للشاشة
onStart	تستدعاء عندما يصبح التطبيق ظاهر للمستخدم
onResume	تستدعاء عندما يستطيع المستخدم التفاعل مع التطبيق
onPause	تستدعاء عندما يكون التطبيق غير ظاهر للمستخدم (مثل ذهاب التطبيق للخلفية او فتح تطبيق اخر)
onStop	تستدعاء عندما التطبيق تأخر بالظهور
onRestart	تستدعاء عندما ترجع للشاشة بعد التوقف
onDestroy	تستدعاء عندما يغلق الشاشة بشكل نهائي

Introduction (Units)

Screen size

Actual physical size, measured as the screen's diagonal. For simplicity, Android groups all actual screen sizes into four generalized sizes: small, normal, large, and extra-large.

Orientation

The orientation of the screen from the user's point of view. This is either landscape or portrait.

Orientation

The orientation of the screen from the user's point of view. This is either landscape or portrait.

PX (Pixels)

Corresponds to actual pixels on the screen.

Screen density

The number of pixels within a physical area of the screen.

Density-independent pixel (dp)

A virtual pixel unit that you should use when defining UI layout.

Unit	Description	Units Per Physical Inch	Density Independent	Same Physical Size On Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density Independent Pixels	~160	Yes	No
sp	Scale Independent Pixels	~160	Yes	No

AndroidManifest.xml

مدير المشروع

يوضع فيه جميع البيانات الاساسية للمشروع، مثل:

- اسم التطبيق
- ايقونة التطبيق
- الاذونات
  - ❖ PHONE_CALL
  - ❖ INTERNET
  - ❖ LOCATION
  - ❖ WIFI
  - ❖ CAMERA
- أسماء الشاشات
- شاشة البداية
- الثيم
- الإصدار المستهدف

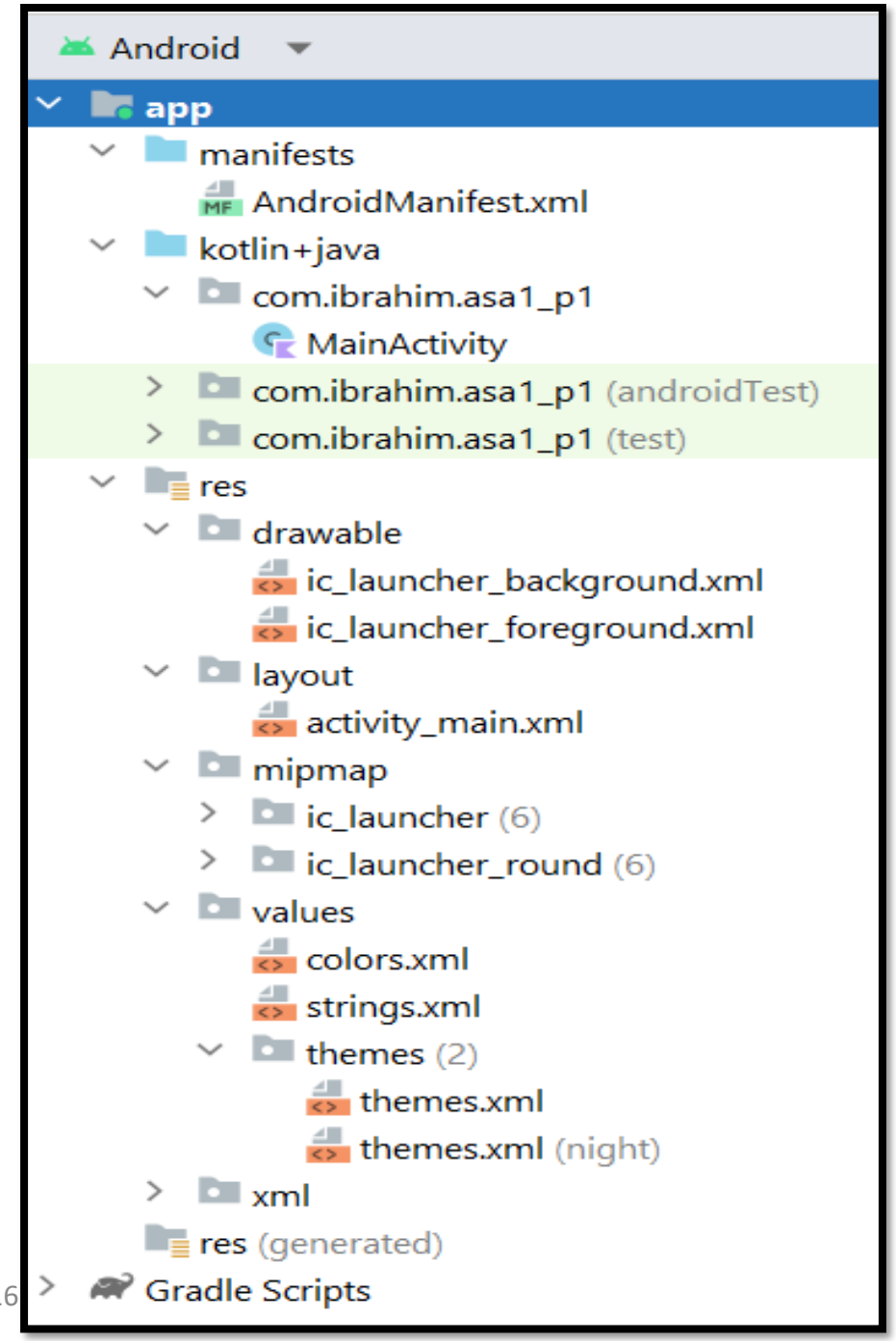
Android

- app
  - manifests
    - AndroidManifest.xml
  - kotlin+java
    - com.ibrahim.asa1_p1
      - MainActivity
    - com.ibrahim.asa1_p1 (androidTest)
    - com.ibrahim.asa1_p1 (test)
  - res
  - res (generated)
- Gradle Scripts

AndroidManifest.xml | MainActivity.kt | activity_main.xml

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="ASA1_P1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ASA1_P1"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

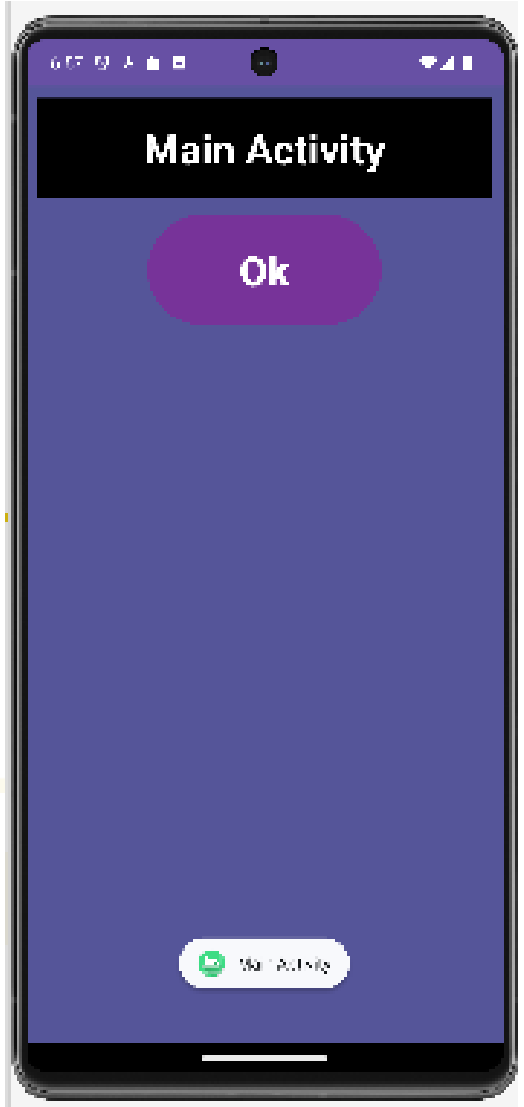
Project


Android
app
manifests
AndroidManifest.xml
kotlin+java
com.ibrahim.asa1_p1
MainActivity
com.ibrahim.asa1_p1 (androidTest)
com.ibrahim.asa1_p1 (test)
res
drawable
ic_launcher_background.xml
ic_launcher_foreground.xml
layout
activity_main.xml
mipmap
ic_launcher (6)
ic_launcher_round (6)
values
colors.xml
strings.xml
themes (2)
themes.xml
themes.xml (night)
xml
res (generated)
Gradle Scripts

Activity_main.xml

MainActivity.kt | activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:background="#555599"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/show"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:background="@color/black"
        android:gravity="center"
        android:padding="20dp"
        android:text="Main Activity"
        android:textColor="@color/white"
        android:textSize="35dp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/ok"
        android:layout_width="201dp"
        android:layout_height="103dp"
        android:layout_margin="10dp"
        android:layout_marginTop="100dp"
        android:backgroundTint="#773399"
        android:gravity="center"
        android:padding="20dp"
        android:text="Ok"
        android:textColor="@color/white"
        android:textSize="35dp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/show" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



MainActivity.kt

MainActivity.kt | activity_main.xml

```kotlin
package com.ibrahim.asa1_p1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val show = findViewById<TextView>(R.id.show)
        var ok = findViewById<Button>(R.id.ok)

        ok.setOnClickListener { it: View!
            Toast.makeText( context: this, show.text.toString(), Toast.LENGTH_LONG).show()
            Toast.makeText( context: this, text: "مرحبا", Toast.LENGTH_LONG).show()
        }
    }
}
```

المكتبات

الملف البرمجي

دالة إنشاء التصميم على شاشة الجوال

جملة الربط بين الملف البرمجي وملف التصميم

جملة الربط بين العناصر في التصميم والملف البرمجي باستخدام id

حدث عند الضغط على زر يقوم بتنفيذ مابين الاقواس

رسالتين تنبيه تضهرا على الشاشة لفترة مؤقته ثم تختفيان تظهر الاولى وبعد ما تنتهي تظهر الثانية

Design for Android

https://developer.android.com/design/ui

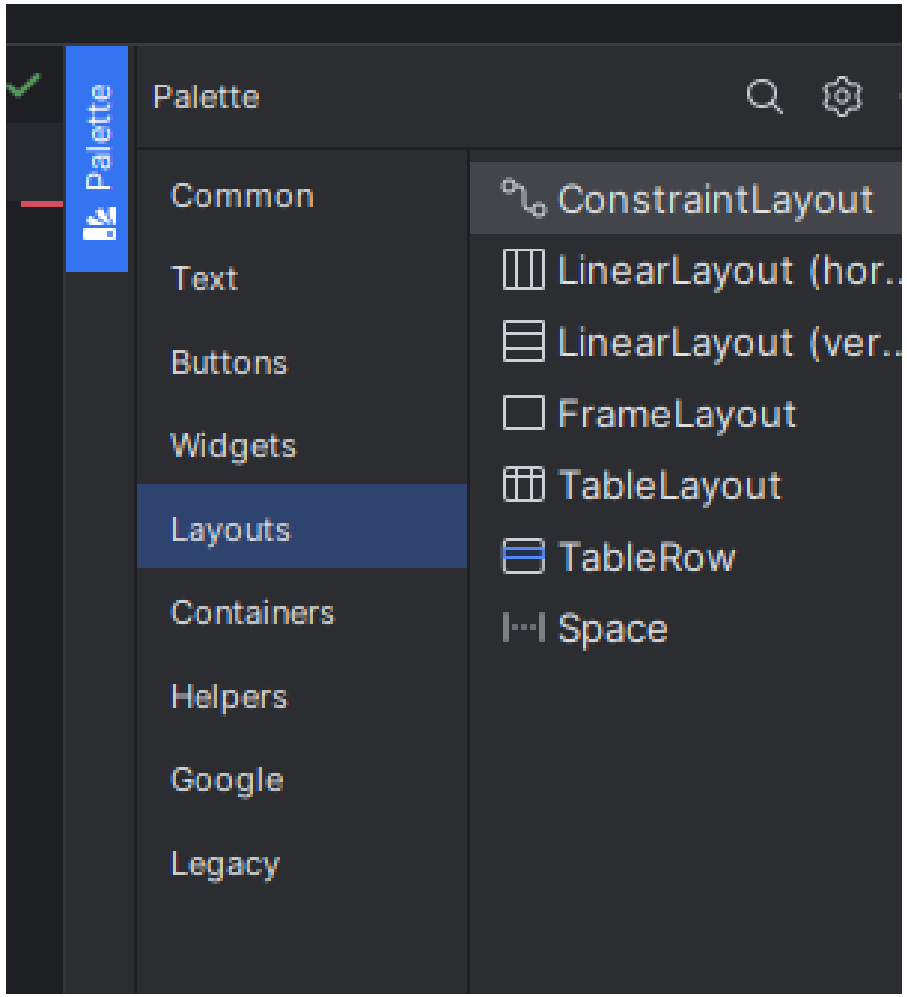
Design beautiful and modern Android apps that meet your user

- where they are,
- whether browsing their phone,
- reading on their tablet,
- glancing at their wrist,
- or watching TV.

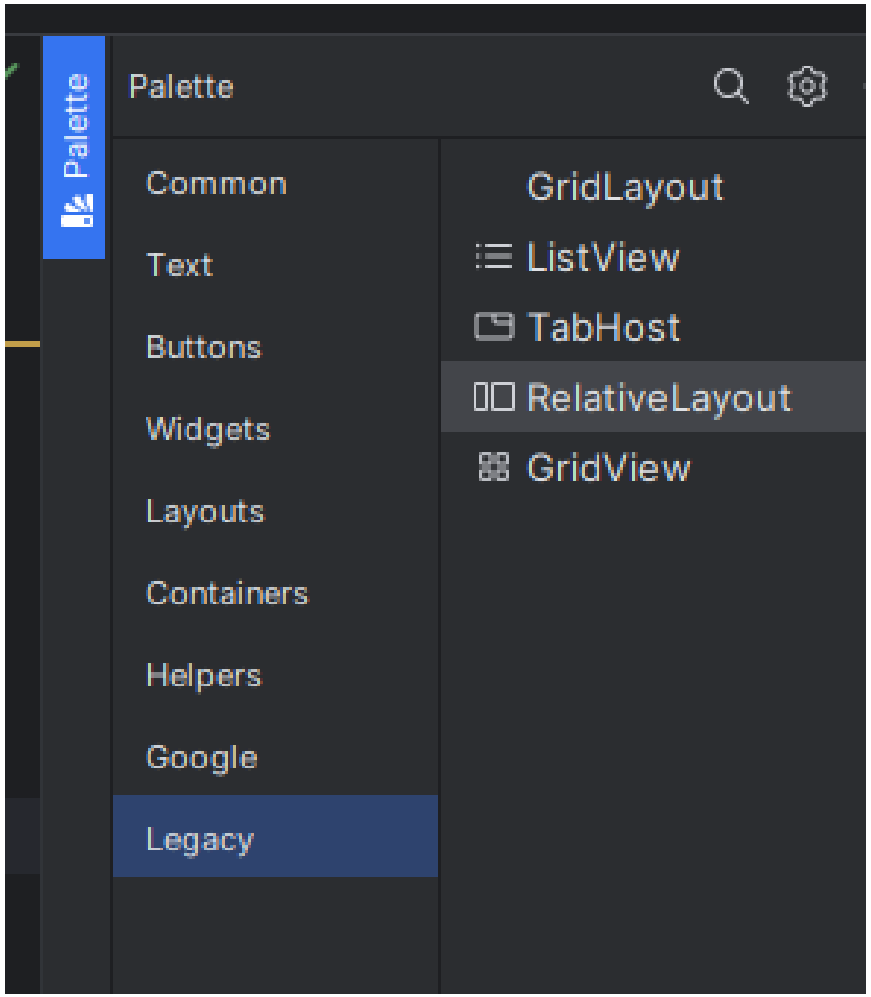
Top 6 Android UI Design Tools in 2024
https://mockitt.com/ui-ux-design/android-ui-design.html

Design for Android

Layout Types in Android



Legacy Types in Android



Design for Android

activity_next.xml

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

```

activity_next.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".NextActivity">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Design for Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Android studio run on physical device

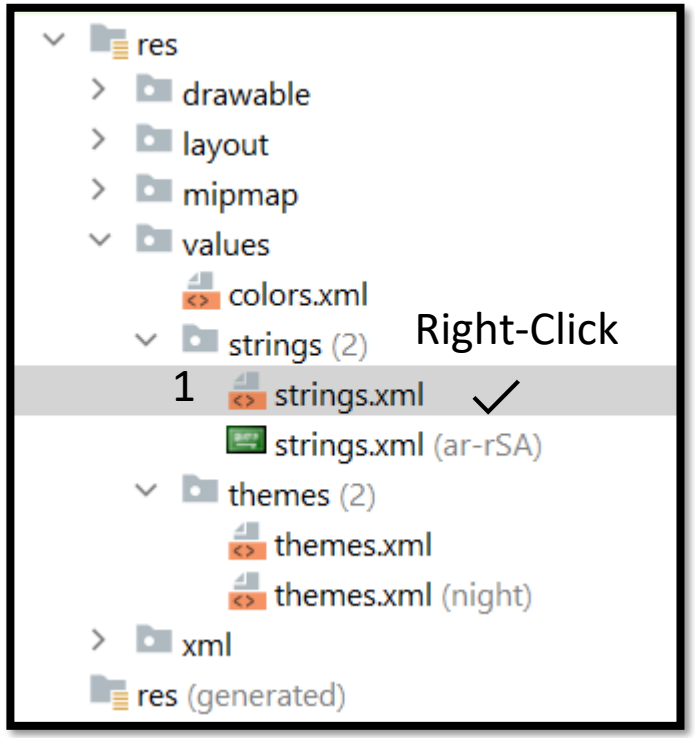
- **Run apps on a hardware device**
- **Set up a device for development**

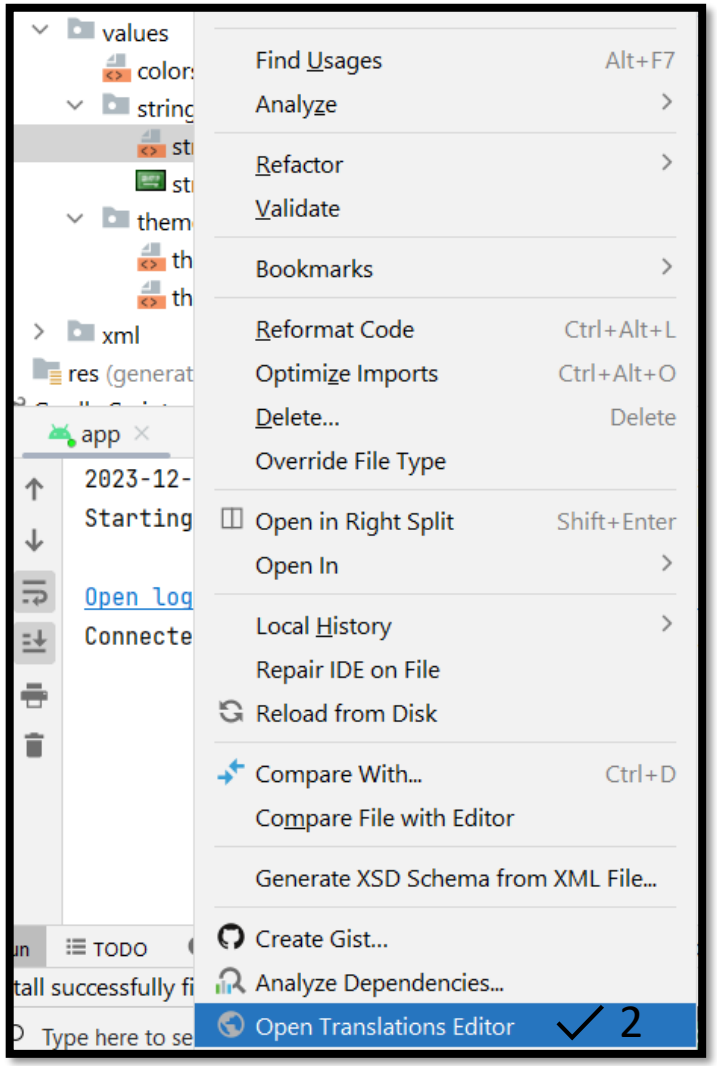
❖ Before you can start debugging on your device, decide whether you want to connect to the device using a USB cable or Wi-Fi. Then do the following:

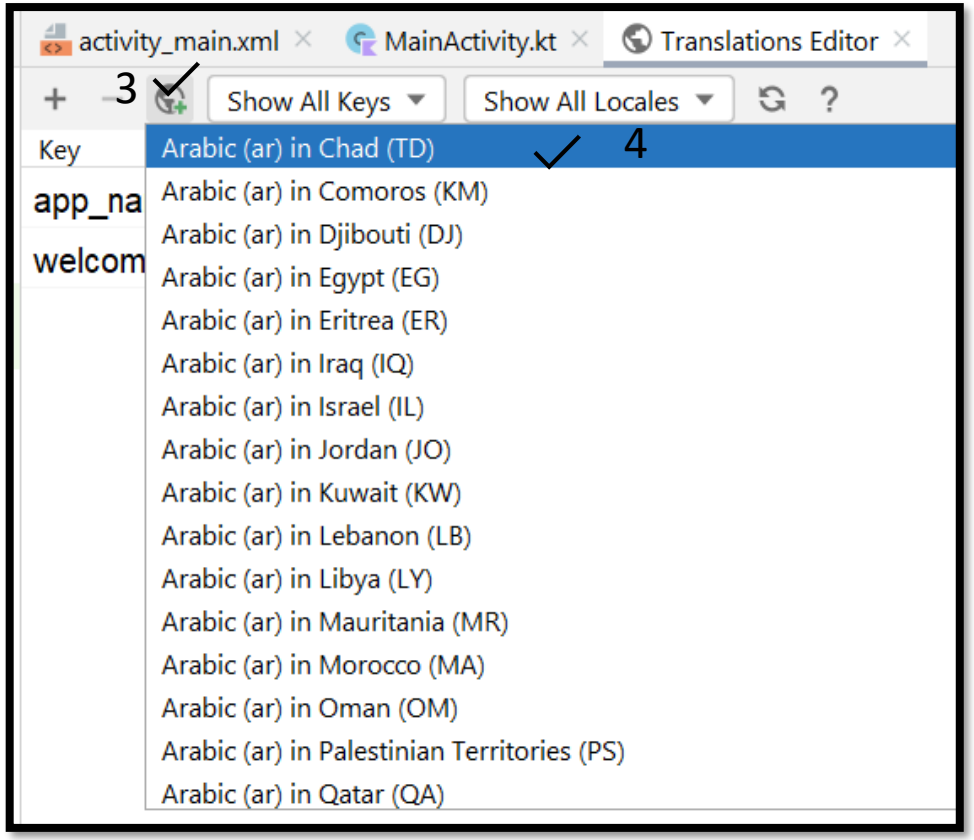
❖ On the device, open the **Settings** app, select **Developer options**, and then enable **USB debugging** (if applicable).

❖ **Windows**: Install a USB driver for ADB (if applicable). For an installation guide and links to OEM drivers, see Install OEM USB drivers.

Localization (التطبيق يعمل بلغة الجهاز)







خطوات أضافة ملف لدعم اللغة العربية

Localization (التطبيق يعمل بلغة الجهاز)

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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:id="@+id/show"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:gravity="center"
        android:text="@string/show"
        android:textSize="36dp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/close"
        android:layout_width="172dp"
        android:layout_height="153dp"
        android:layout_marginTop="96dp"
        android:text="@string/close"
        android:textSize="30dp"
        android:textStyle="italic"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/show" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

لغة التطبيق تعمل بلغة الجهاز

activity_main.xml | strings.xml

Edit translations for all locales in the translations editor.

```xml
<resources>
    <string name="app_name">ASA1_Icon</string>
    <string name="show">show</string>
    <string name="close">close</string>
</resources>
```

activity_main.xml | values\strings.xml | ar-rSA\strings.xml

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)

Edit translations for all locales in the translations editor.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">تصميم ايقونة</string>
    <string name="show">عرض</string>
    <string name="close">أغلاق</string>
</resources>
```

Full Screen [NoActionBar – No Title]

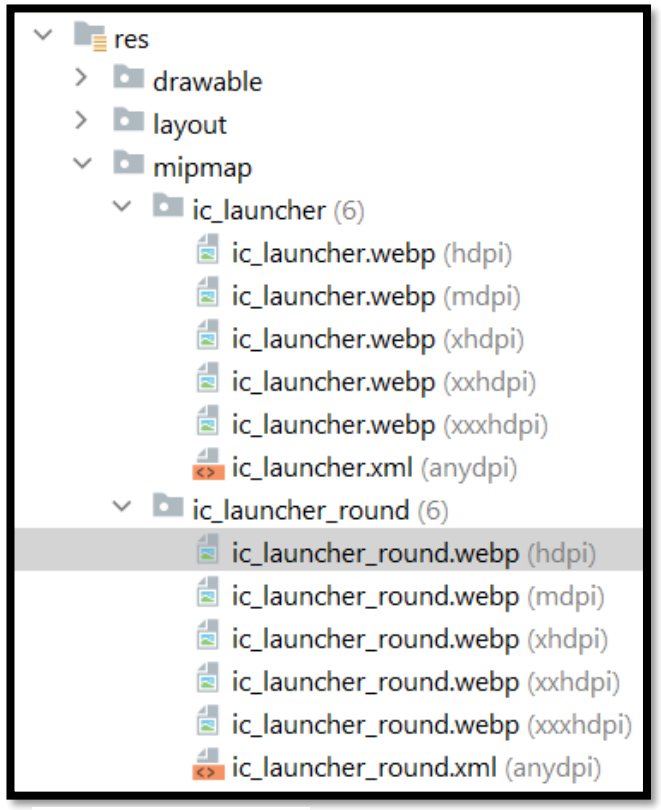
MainActivity.kt | activity_main.xml | themes.xml

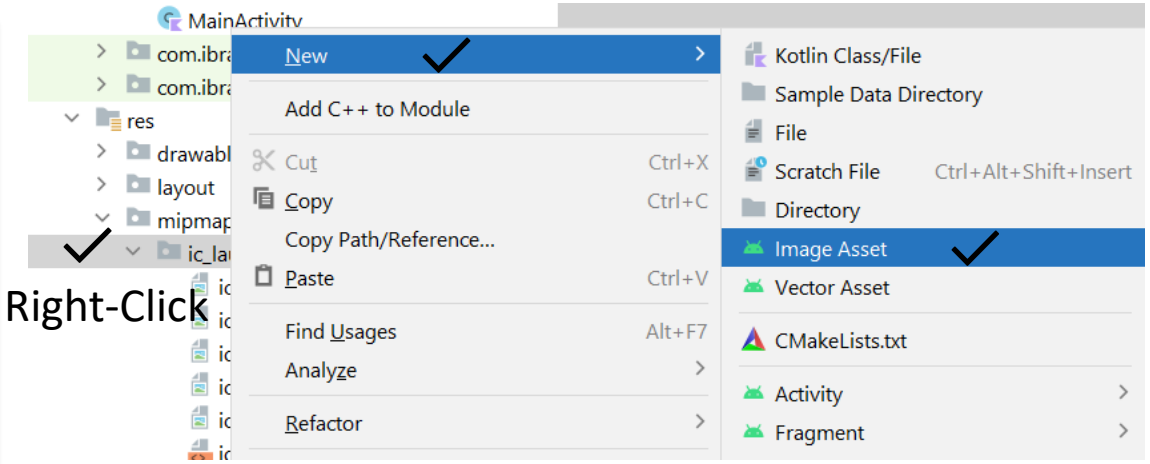
```kotlin
package com.ibrahim.asa_localization
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Window
import android.view.WindowManager
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Hide Title
        getWindow().requestFeature(Window.FEATURE_NO_TITLE)
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN)
        setContentView(R.layout.activity_main)
    }
}
```

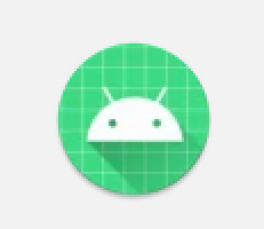
MainActivity.kt | activity_main.xml | themes.xml

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Base.Theme.ASA_Localization" parent="Theme.Material3.DayNight.NoActionBar">
    </style>
    <style name="Theme.ASA_Localization" parent="Base.Theme.ASA_Localization" />
</resources>
```

Create App Icon



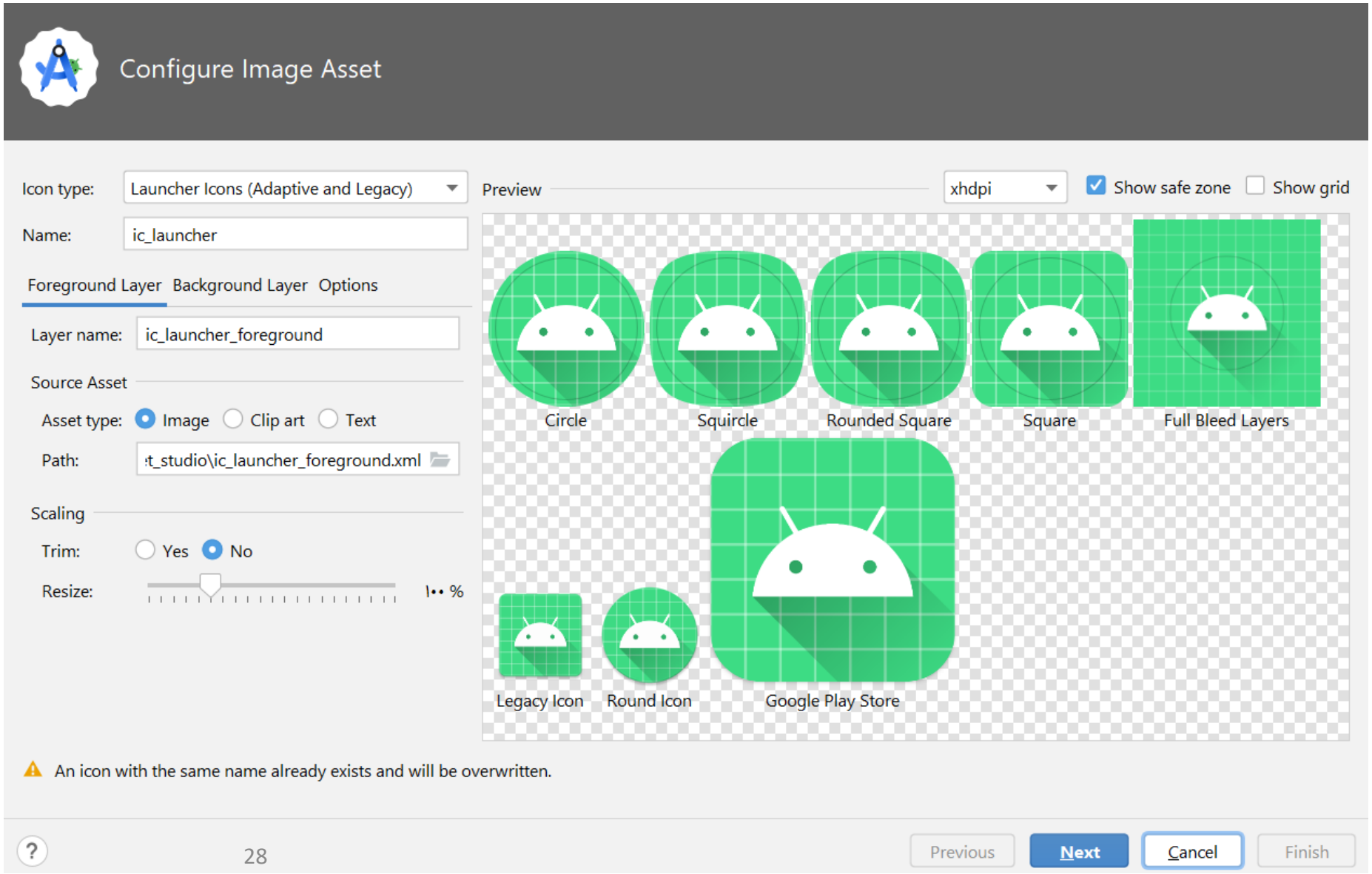




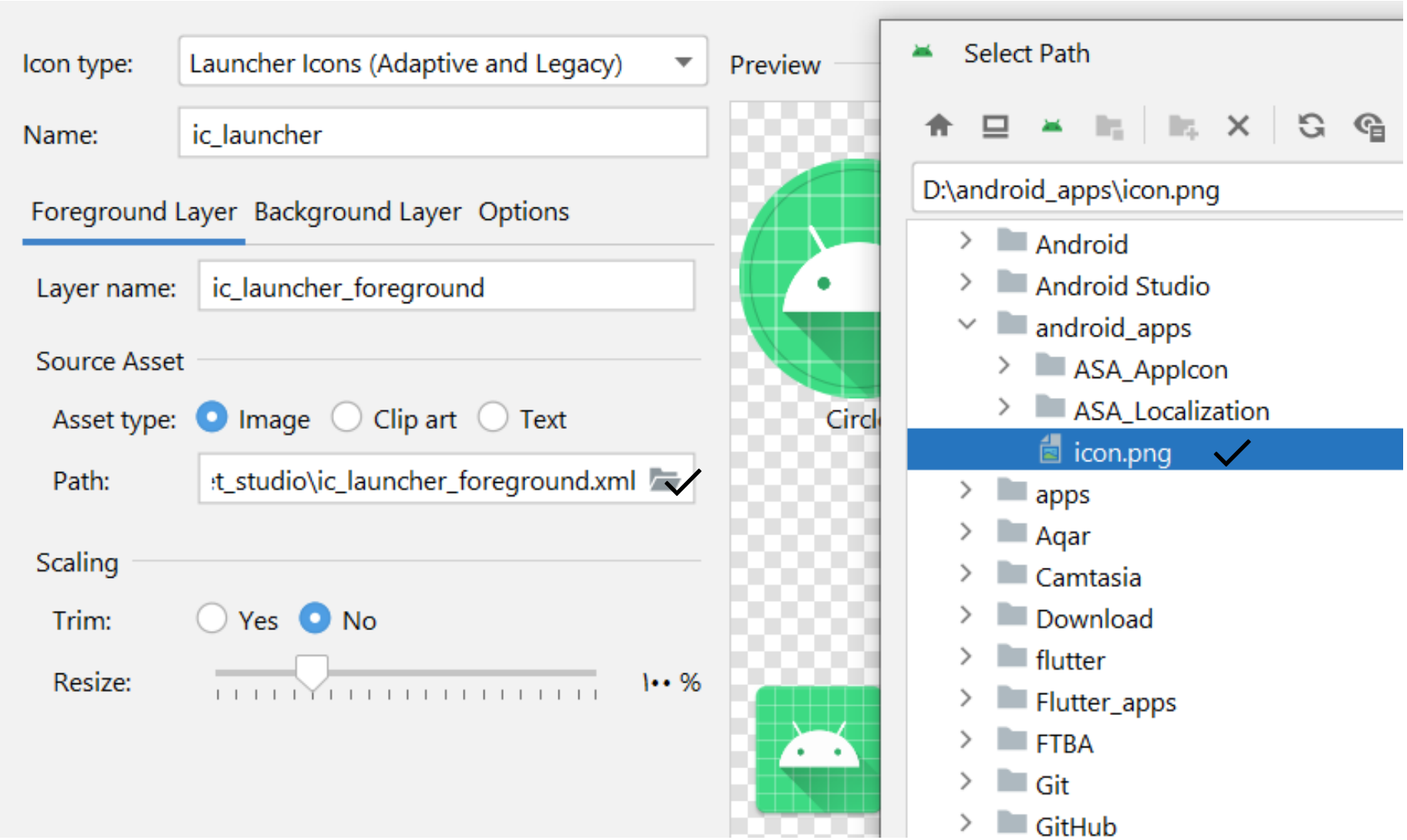
Google Play icon design specifications

https://developer.android.com/distribute/google-play/resources/icon-design-specifications

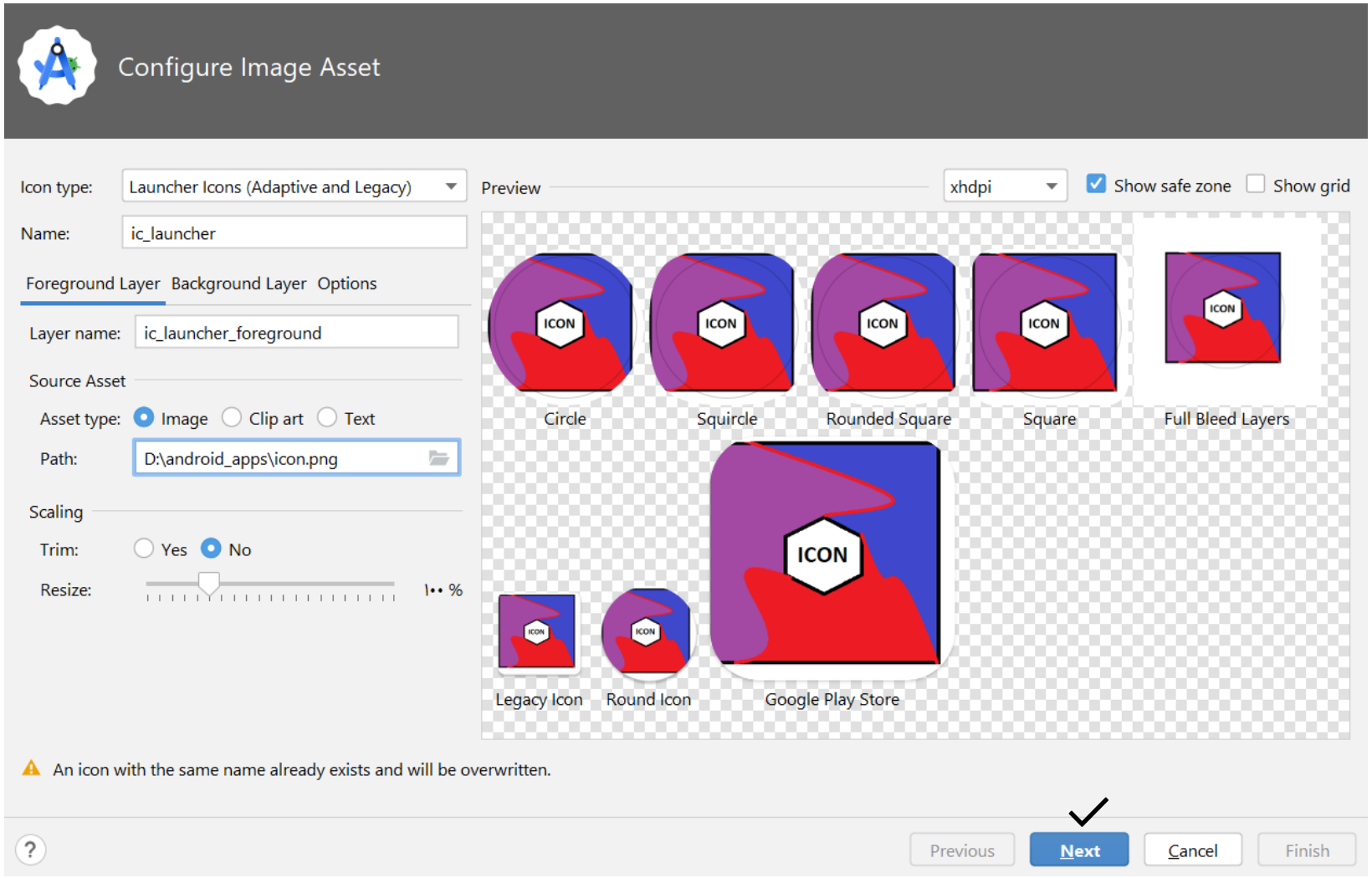
Create App Icon … Cont



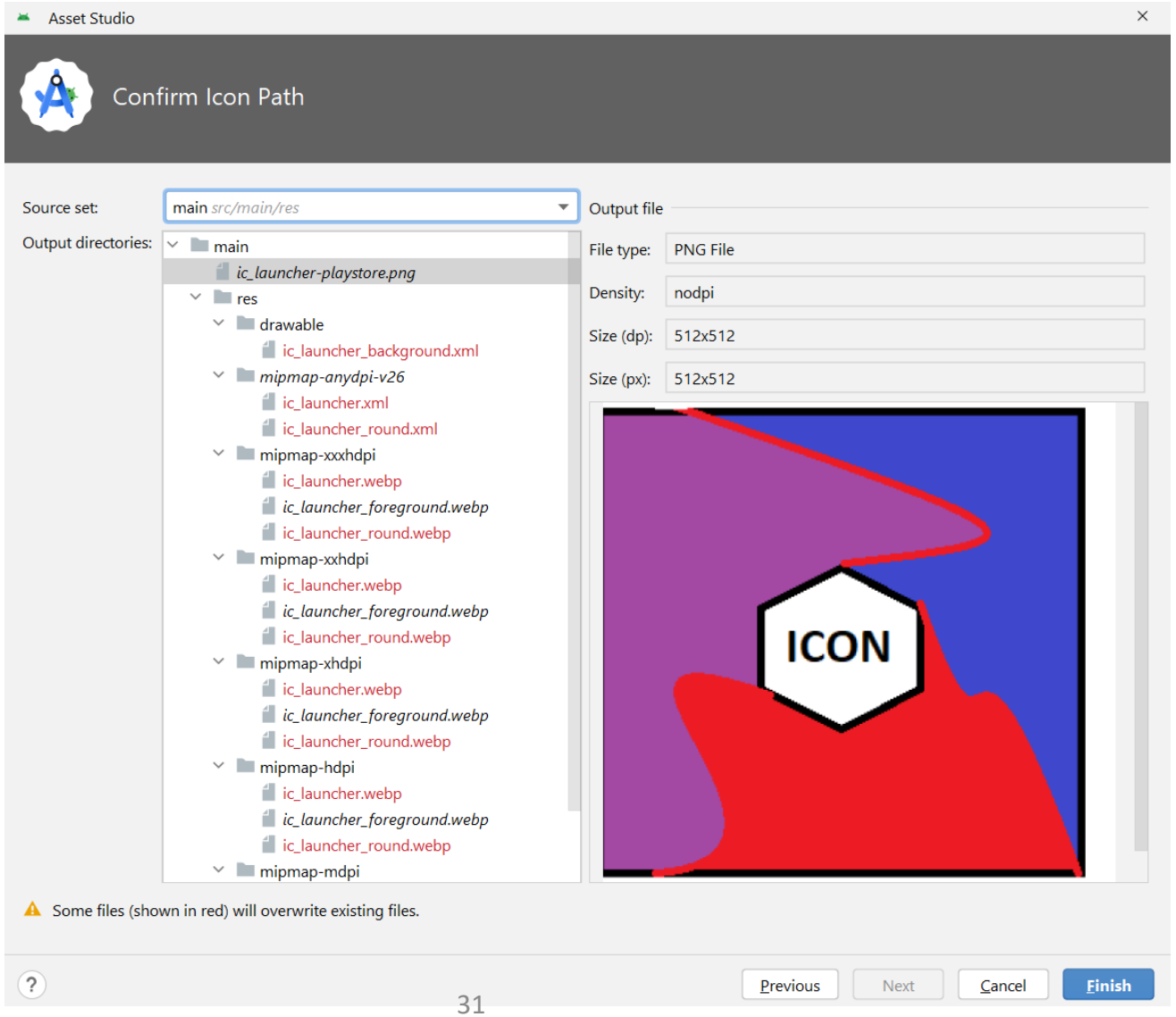
Create App Icon … Cont

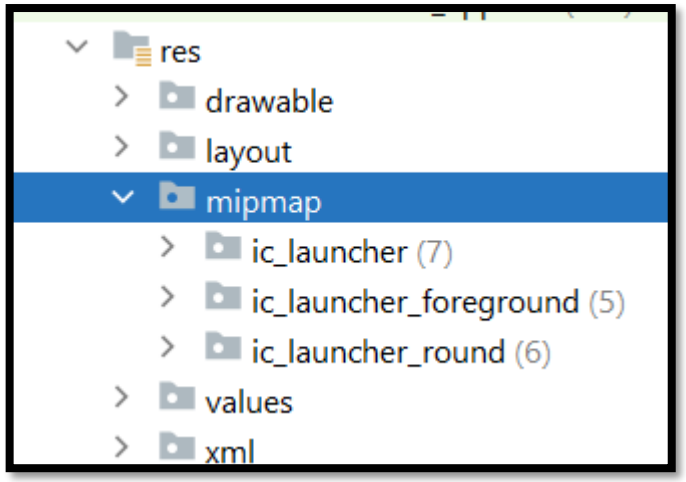


Create App Icon … Cont



Create App Icon … Cont





Create App Icon … Cont

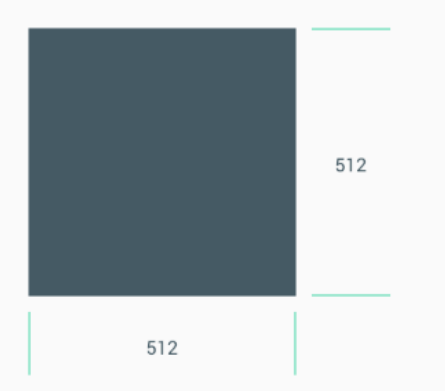
What is WebP in Android Studio?

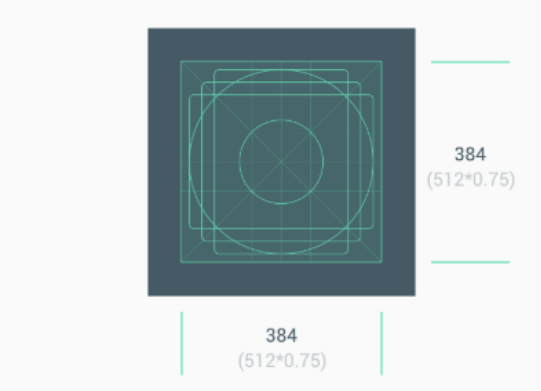
WebP is an image file format from Google that provides lossy compression (like JPEG) as well as transparency (like PNG) but can provide better compression than either JPEG or PNG.

Images

You can import your own images and adjust them for the icon type. Image Asset Studio supports the following file types: PNG (preferred), JPG (acceptable), and GIF (discouraged).

Image Asset Studio places the icons in the proper locations in the res/mipmap-*density*/ directories.
It also creates a 512 x 512 pixel image that's appropriate for the Google Play store.



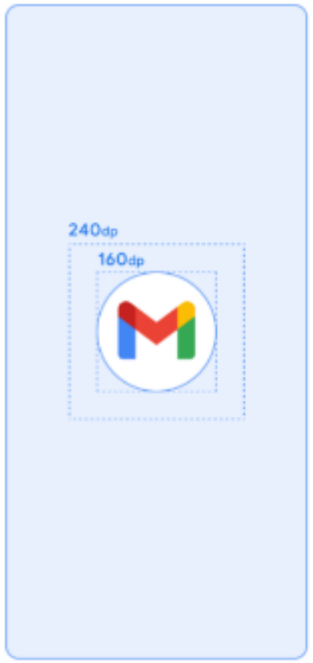


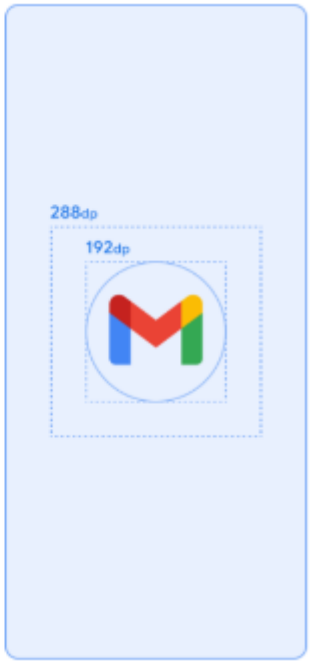
Splash Screen

Splash screen dimensions

The splash screen icon uses the same specifications as adaptive icons, as follows:

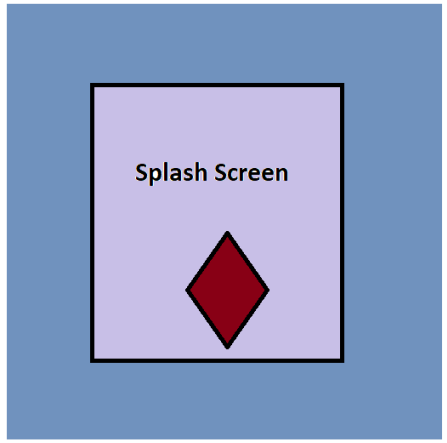
- Branded image: this must be 200×80 dp.
- App icon with an icon background: this must be 240×240 dp and fit within a circle 160 dp in diameter.
- App icon without an icon background: this must be 288×288 dp and fit within a circle 192 dp in diameter.
- For example, if the full size of an image is 300×300 dp, the icon needs to fit within a circle with a diameter of 200 dp. Everything outside the circle turns invisible (masked).
- 1/3 - 2/3

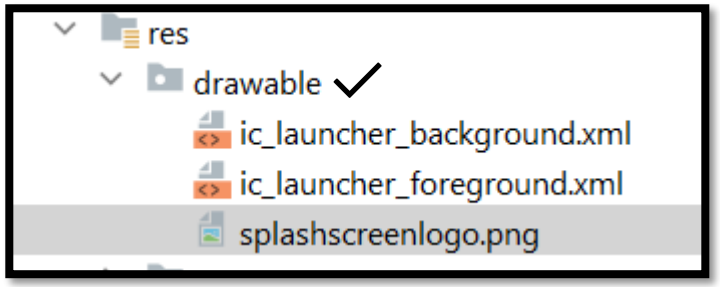




Splash Screen

Step 1: Design logo then **Copy/Paste** into **<u>drawable</u>** folder





Splash Screen

Step 2: Add the following snippet to your **build.gradle** file

https://developer.android.com/develop/ui/views/launch/splash-screen#kts

Get started

The core `SplashScreen` library brings the Android 12 splash screen to all devices from API 23. To add it to your project, add the following snippet to your `build.gradle` file:

Groovy | Kotlin

```
dependencies {
    implementation("androidx.core:core-splashscreen:1.0.0")
}
```

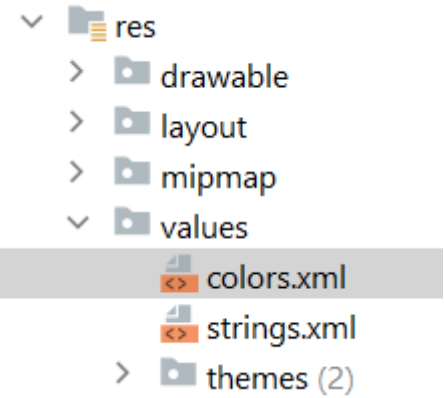
build.gradle.kts (:app)

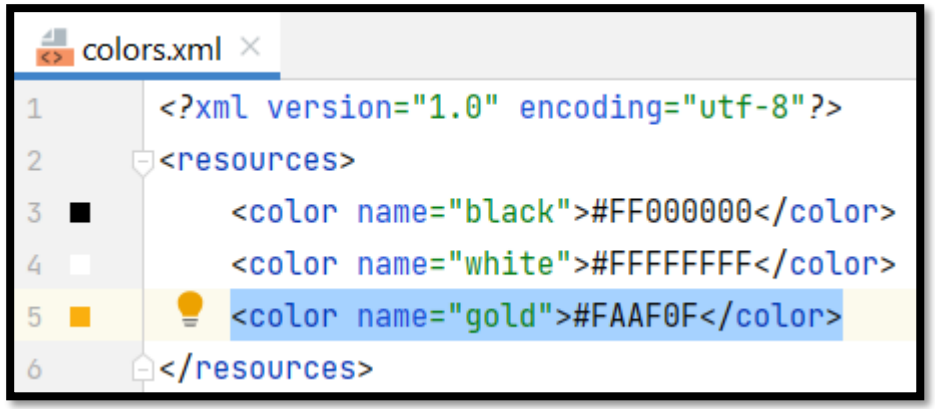
Gradle files have changed since last project sync. A project sync may be necess... Sync Now Ignore these changes

```
    }
}

dependencies { this: DependencyHandlerScope

    implementation("androidx.core:core-ktx:1.12.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
    implementation ("androidx.core:core-splashscreen:1.0.0")
}
```

Splash Screen

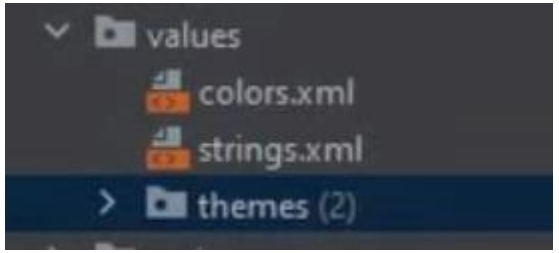
Step 3: Add background color





Splash Screen

Step 4: Add Theme to themes.xml file

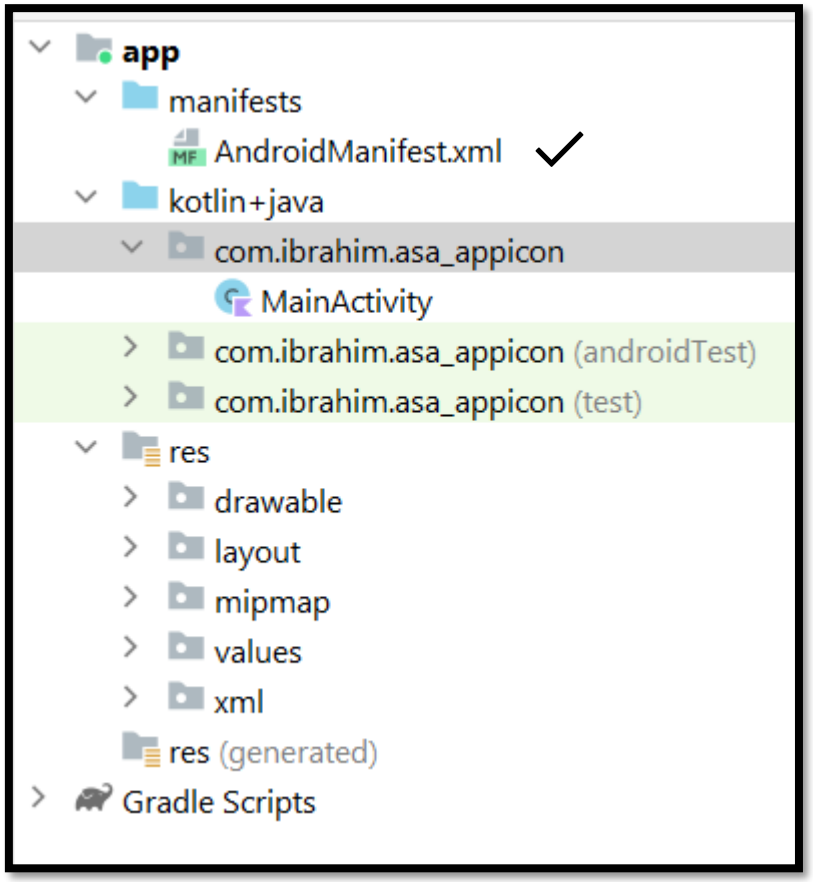


themes.xml

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <style name="Theme.ASA_AppIcon" parent="Base.Theme.ASA_AppIcon" />

    <!-- Base application theme. -->
    <style name="Base.Theme.SplashScreenLogo" parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Customize your light theme here. -->
        <item name="colorPrimary">@color/gold</item>
        <item name="colorPrimaryVariant">@color/gold</item>
        <item name="colorOnPrimary">@color/gold</item>
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
    </style>

    <style name="Theme.SplashScreenLogo" parent="Base.Theme.SplashScreenLogo" />

    <style name="Theme.App.SplashScreen" parent="Theme.SplashScreen">
        <item name="windowSplashScreenBackground">@color/gold</item>
        <item name="windowSplashScreenAnimatedIcon">@drawable/splashscreenlogo</item>
        <item name="postSplashScreenTheme">@style/Base.Theme.SplashScreenLogo</item>
    </style>
</resources>
```

Splash Screen

Step 5: Add Theme to AndroidManifest.xml file



themes.xml | AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="ASA_AppIcon"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SplashScreenLogo"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:theme="@style/Theme.App.SplashScreen"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Splash Screen

Step 6: Change MainActivity

MainActivity.kt

```kotlin
package com.ibrahim.asa_appicon

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.splashscreen.SplashScreen.Companion.installSplashScreen

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Thread.sleep( millis: 3000)
        installSplashScreen()
        setContentView(R.layout.activity_main)
    }
}
```

Android Animation

Animation in android apps is:

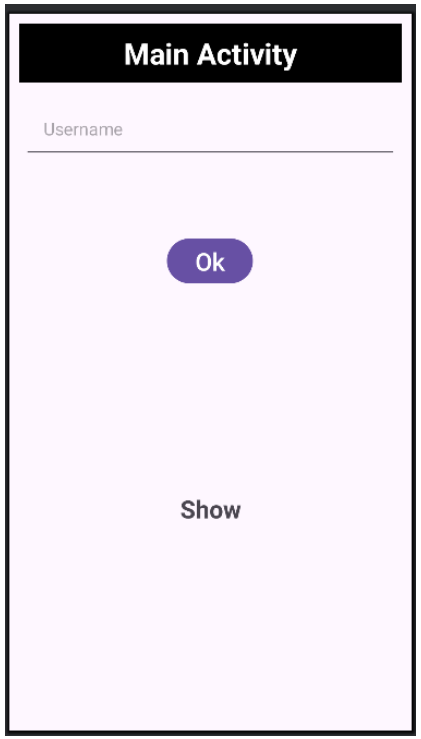
- the process of creating motion
- and shape change.

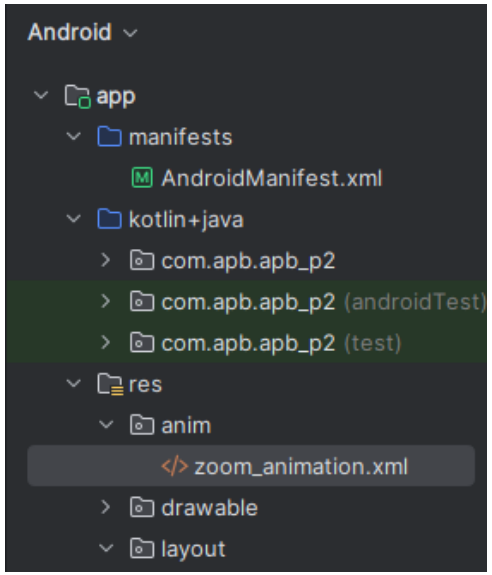
https://github.com/idekhail/APA_Animation1

The basic ways of animation that we'll look upon in this tutorial are:

1. Fade In Animation
2. Fade Out Animation
3. Cross Fading Animation
4. Blink Animation
5. Zoom In Animation
6. Zoom Out Animation
7. Rotate Animation
8. Move Animation
9. Slide Up Animation
10. Slide Down Animation
11. Bounce Animation
12. Sequential Animation
13. Together Animation

Animation





MainActivity.kt | zoom_animation.xml | activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true" >

    <scale
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:duration = "1000"
        android:fromXScale = "0"
        android:fromYScale = "0"
        android:pivotX = "50%"
        android:pivotY = "50%"
        android:toXScale = "1"
        android:toYScale = "1"/>
</set>
```

https://github.com/idekhail/APB_P2-Animation-EditText
https://www.geeksforgeeks.org/implement-zoom-in-or-zoom-out-in-android/?ref=ml_lbp

Exam Refrence

- https://www.interviewbit.com/android-mcq/
- https://www.javaguides.net/2023/12/android-quiz-mcq-questions-and-answers.html