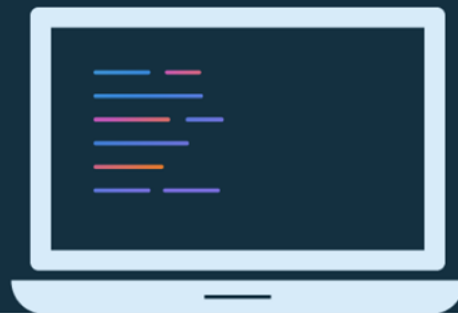




# Bài học 4: Xây dựng ứng dụng Android đầu tiên



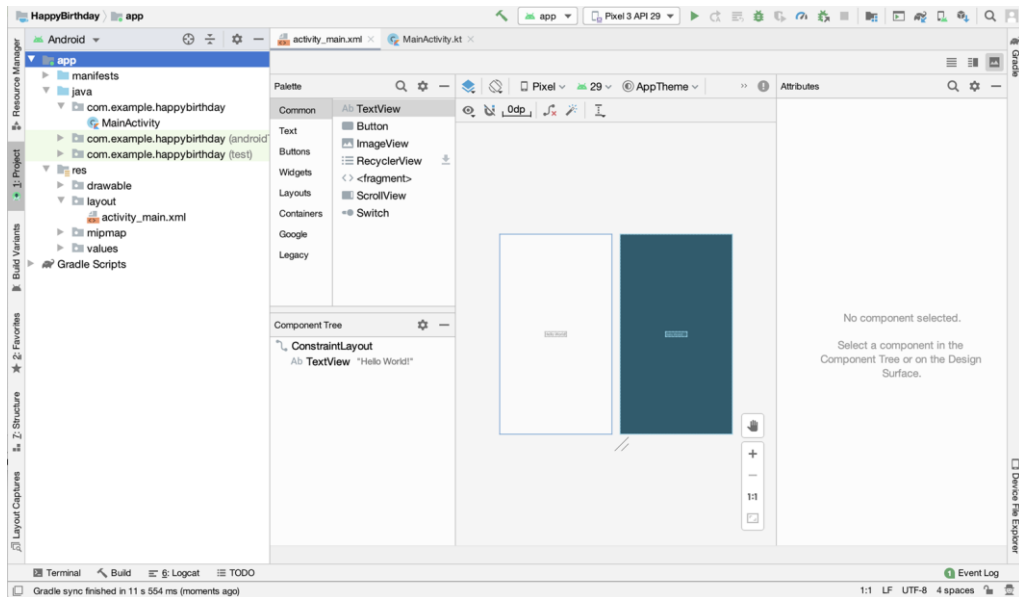
# Giới thiệu về bài học này

## Bài học 4: Xây dựng ứng dụng Android đầu tiên

- [Ứng dụng đầu tiên của bạn](#)
- [Phân tích các thành phần của một ứng dụng Android](#)
- [Bố cục và tài nguyên trong Android](#)
- [Hoạt động](#)
- [Tạo một ứng dụng giàu tính tương tác](#)
- [Gradle: Xây dựng một ứng dụng Android](#)
- [Hỗ trợ tiếp cận](#)
- [Tóm tắt](#)

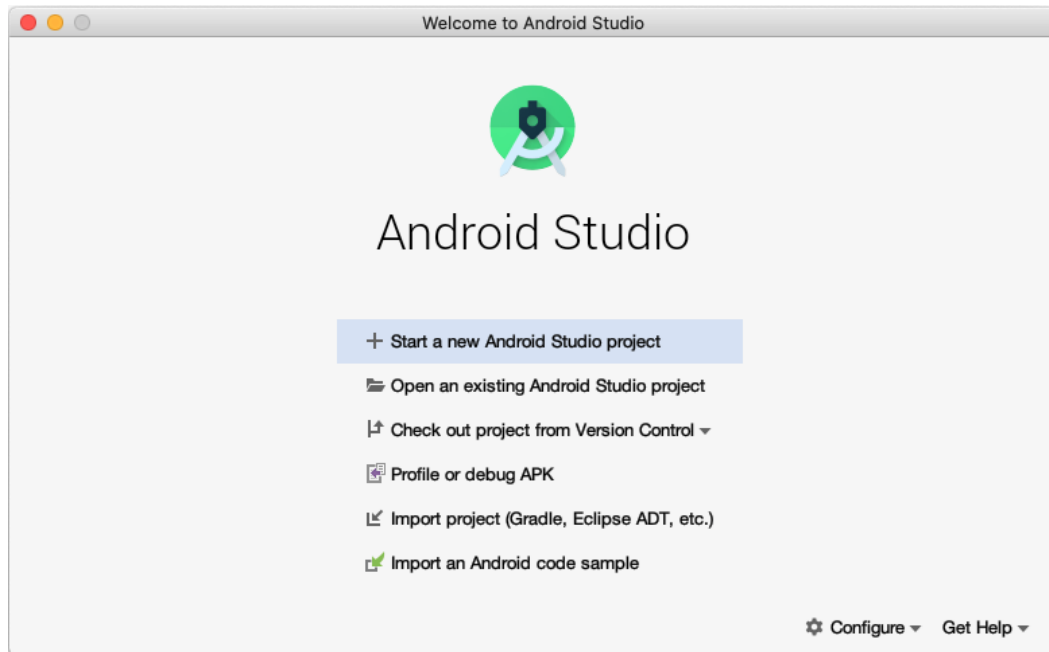
# Android Studio

IDE chính thức để xây dựng ứng dụng Android

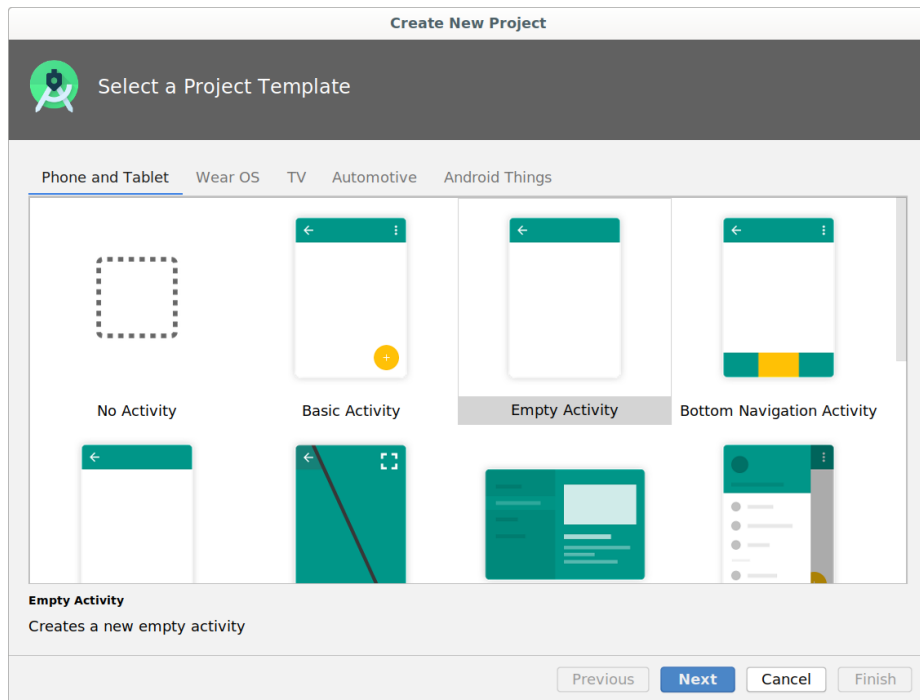


# Ứng dụng đầu tiên của bạn

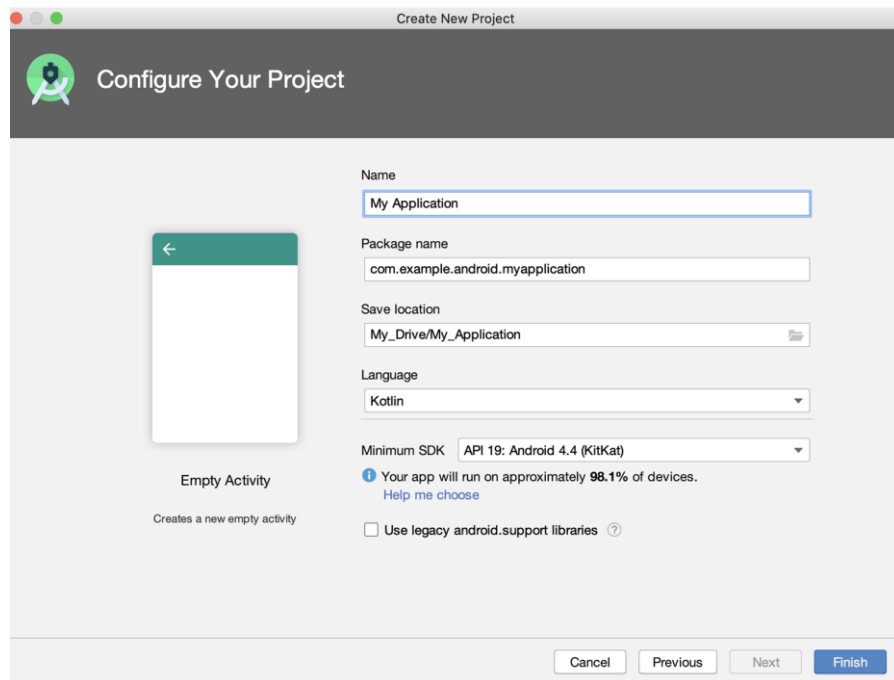
# Mở Android Studio




# Tạo dự án mới

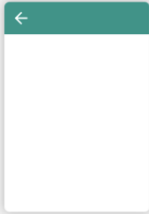


# Nhập thông tin chi tiết về dự án



Create New Project

 Configure Your Project



Empty Activity  
Creates a new empty activity


Name


Package name

Save location

Language

Minimum SDK

 Your app will run on approximately 98.1% of devices.  
[Help me choose](#)

☐ Use legacy android.support libraries 

# Các bản phát hành Android và cấp độ API

Platform Version	API Level	VERSION_CODE
Android 10.0	29	Q
Android 9	28	P
Android 8.1	27	O_MR1
Android 8.0	26	O
Android 7.1.1 Android 7.1	25	N_MR1
Android 7.0	24	N
Android 6.0	23	M
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP



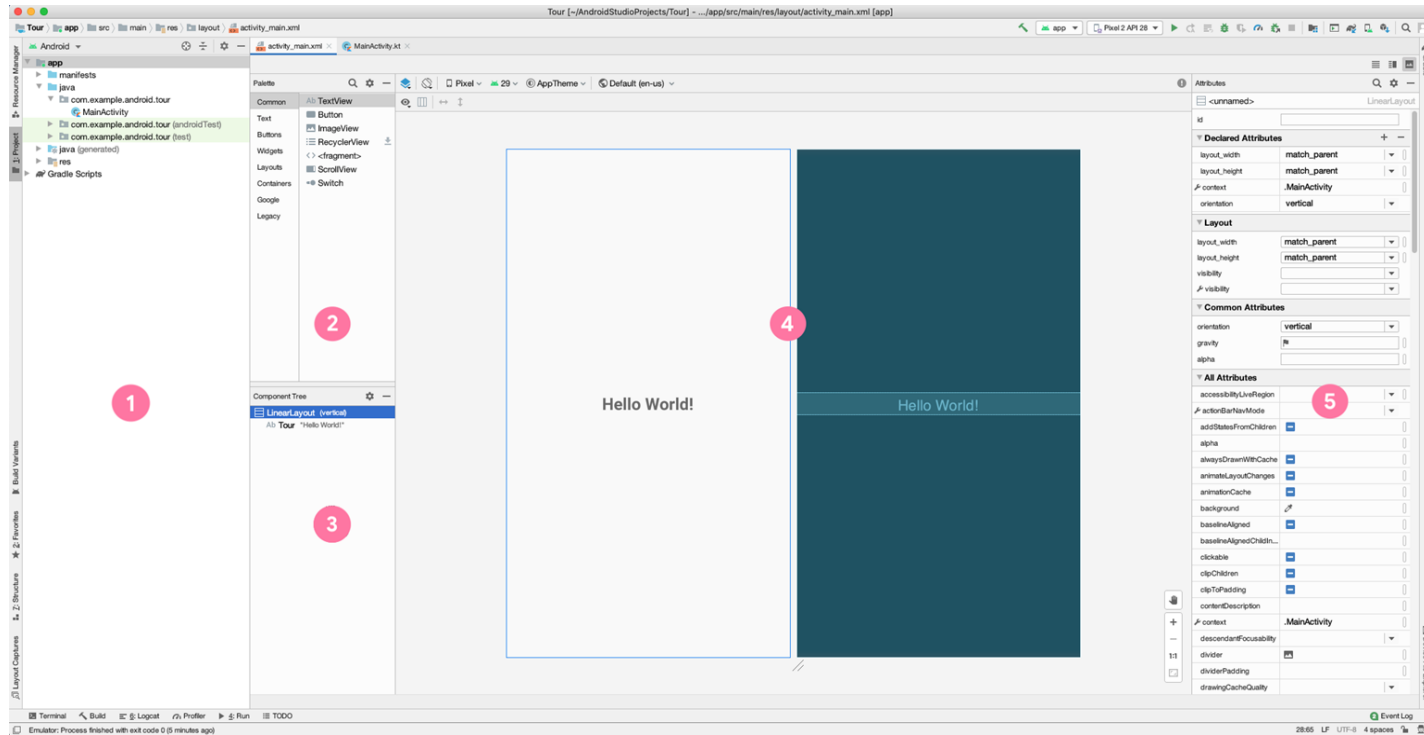
# Chọn các cấp độ API cho ứng dụng của bạn

- SDK tối thiểu: Thiết bị cần có tối thiểu cấp độ API này để cài đặt
- SDK mục tiêu: Phiên bản API và phiên bản Android cao nhất được kiểm tra
- SDK biên dịch: SDK dùng để biên dịch phiên bản thư viện hệ điều hành Android

```
minSdkVersion <= targetSdkVersion <= compileSdkVersion
```

Cấp độ API xác định phiên bản API khung của SDK Android.

# Hướng dẫn về Android Studio

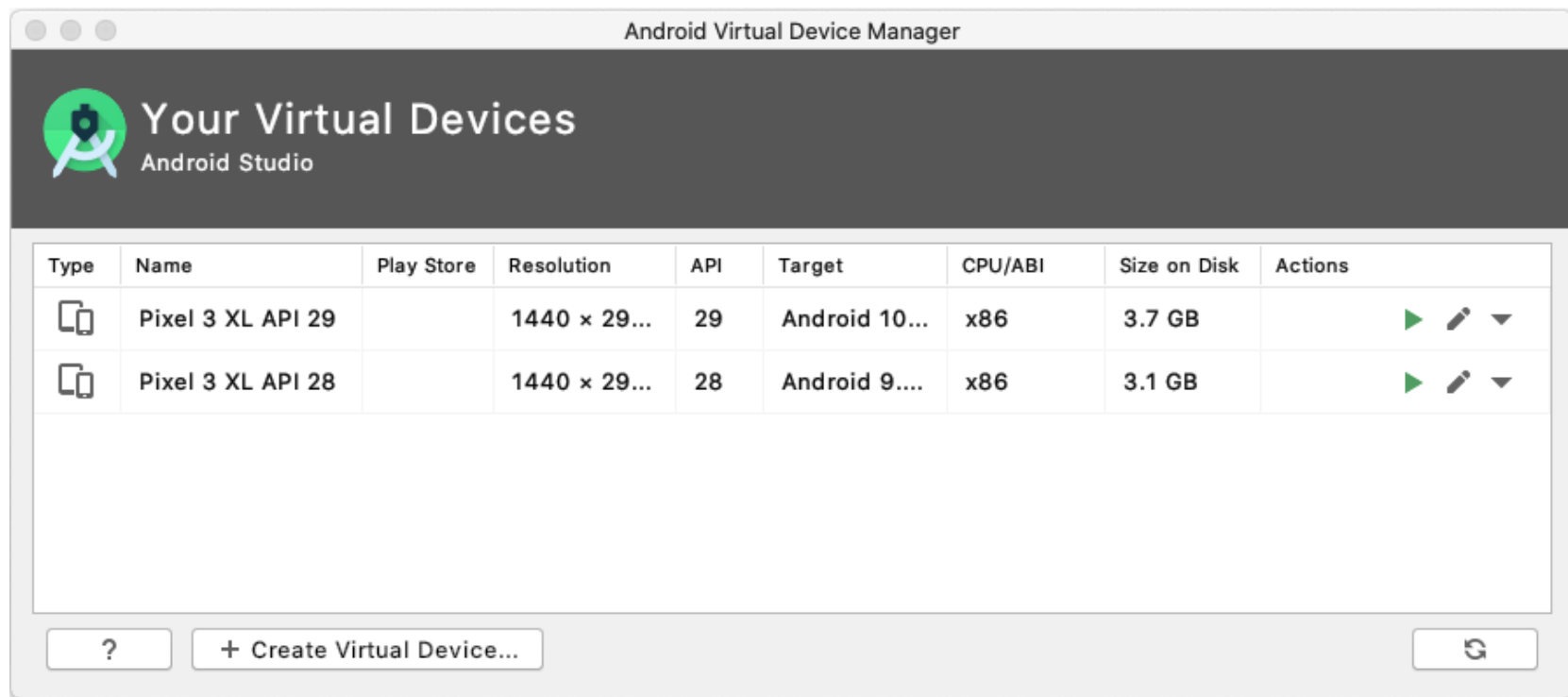


# Chạy ứng dụng của bạn



- Thiết bị Android (điện thoại, máy tính bảng)
- Trình mô phỏng trên máy tính

# Trình quản lý thiết bị ảo Android (AVD)

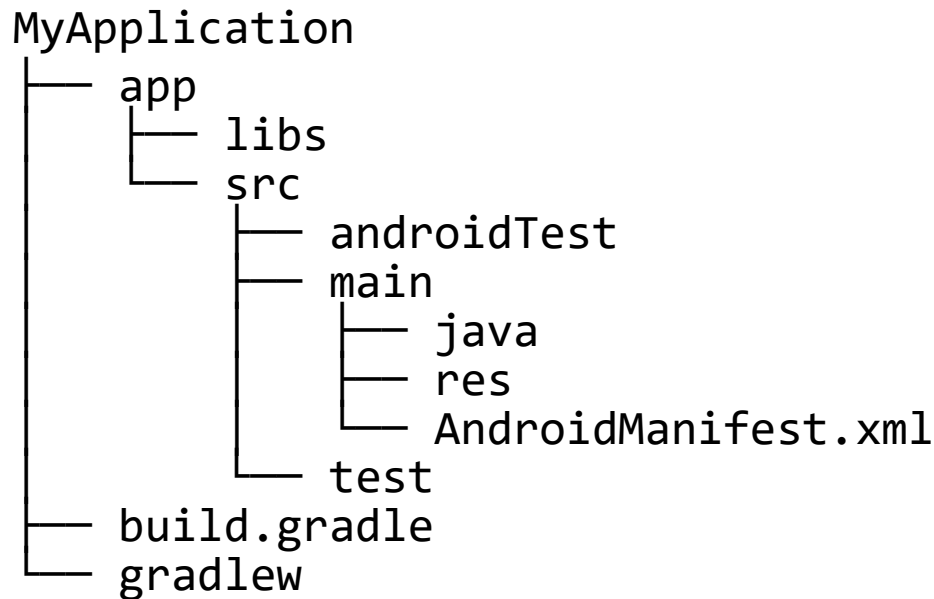


# Phân tích các thành phần của một dự án ứng dụng Android

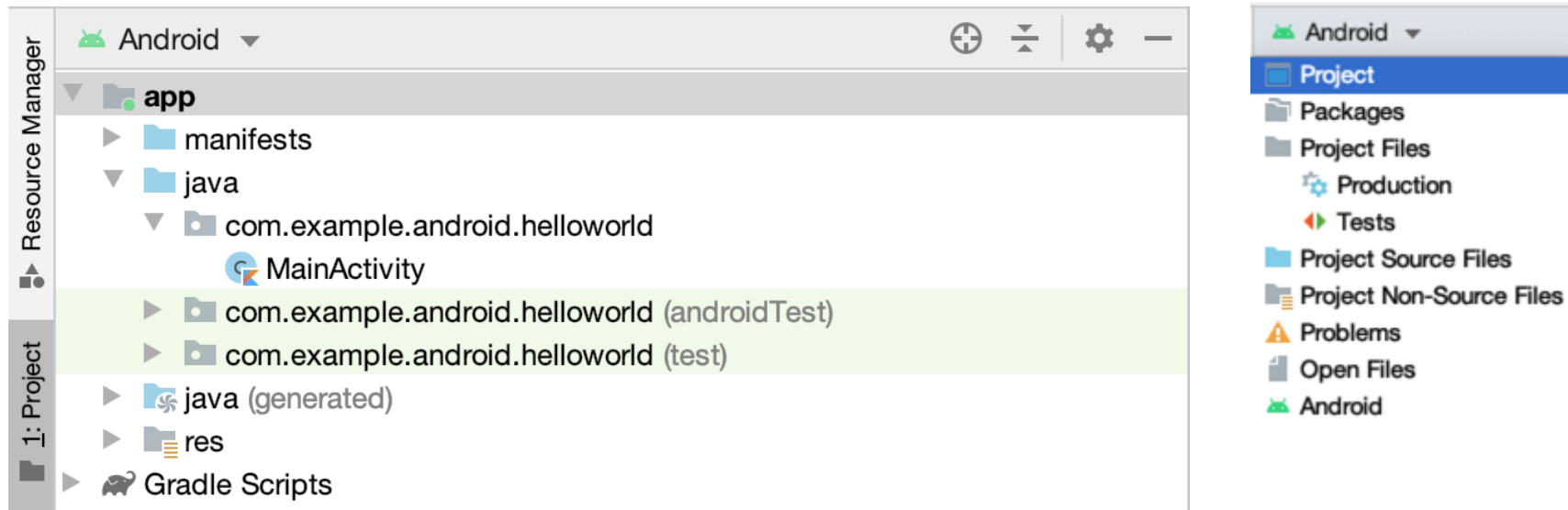
# Phân tích các thành phần của một dự án ứng dụng cơ bản

- Hoạt động
- Tài nguyên (tệp bố cục, hình ảnh, tệp âm thanh, giao diện và màu sắc)
- Tập Gradle

# Cấu trúc của dự án ứng dụng Android



# Duyệt xem tệp trong Android Studio



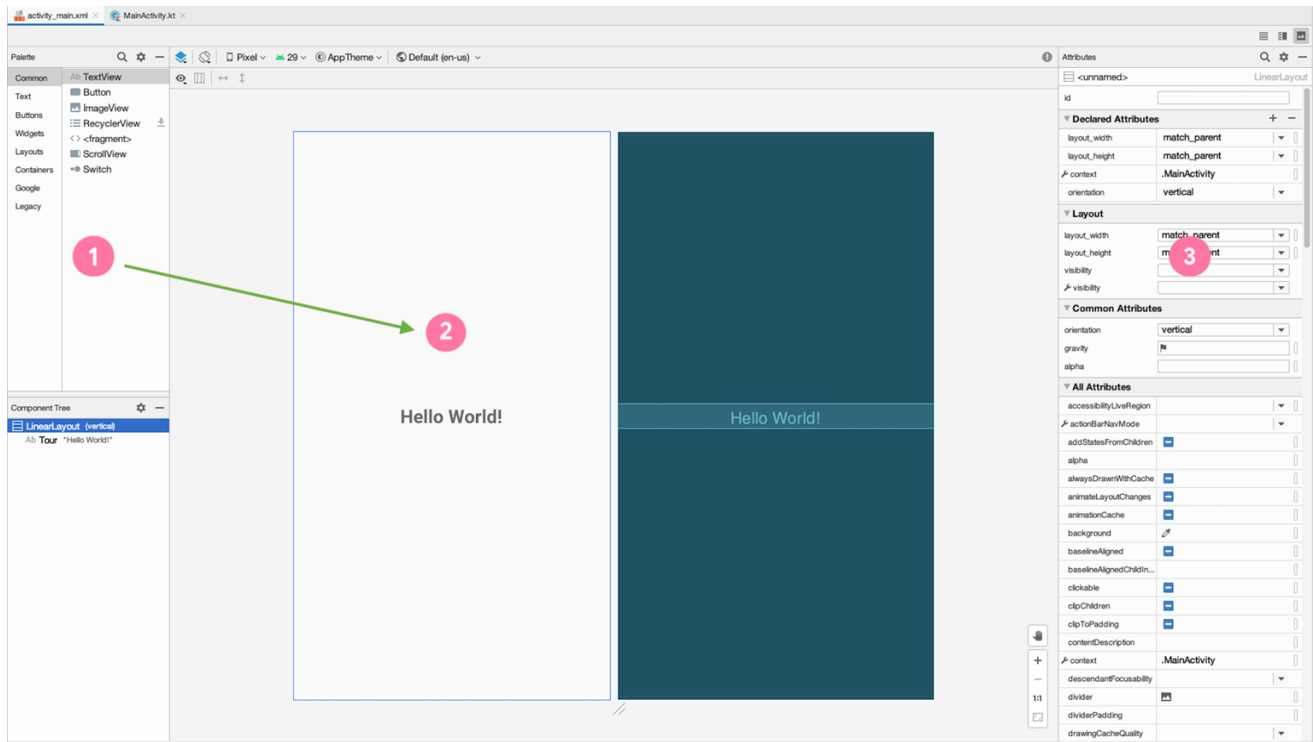


# Bố cục và tài nguyên trong Android

# Chế độ xem

- Chế độ xem là các khối tạo nên giao diện người dùng trong Android
  - Được vạch ranh giới bằng một khu vực hình chữ nhật trên màn hình
  - Chịu trách nhiệm vẽ và xử lý sự kiện
  - Ví dụ: Chế độ xem văn bản, Chế độ xem hình ảnh, Nút
- Có thể được nhóm lại để tạo thành các giao diện người dùng phức tạp hơn

# Layout Editor



# Bố cục XML

Bạn cũng có thể chỉnh sửa bố cục của mình trong tệp XML.

- Android dùng tệp XML để chỉ định bố cục của giao diện người dùng (bao gồm cả thuộc tính của Chế độ xem)
- Mỗi Chế độ xem trong tệp XML tương ứng với một lớp trong Kotlin giúp kiểm soát cách hoạt động của Chế độ xem đó

# Tệp XML cho Chế độ xem văn bản

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World!"/>
```

Xin chào!

# Kích thước của Chế độ xem

- wrap\_content

```
android:layout_width="wrap_content"
```

- match\_parent

```
android:layout_width="match_parent"
```

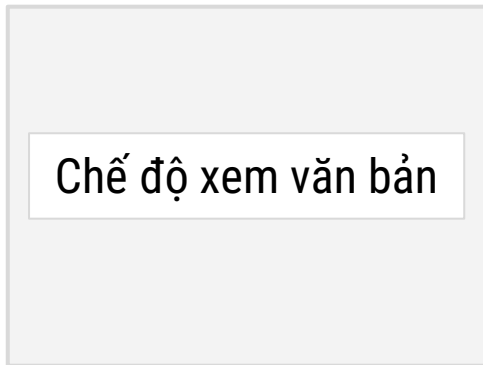
- Fixed value (use dp units)

```
android:layout_width="48dp"
```

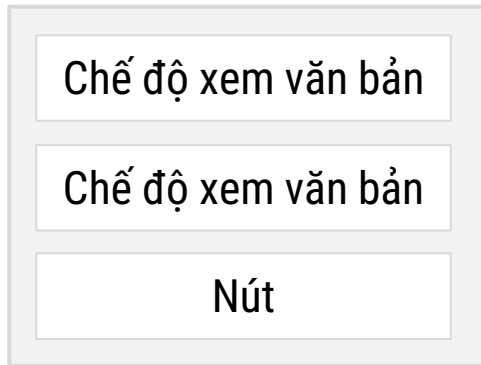
# ViewGroup

ViewGroup là một vùng chứa xác định cách hiển thị của các chế độ xem.

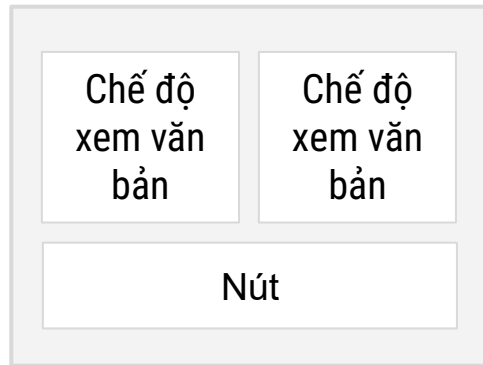
FrameLayout



LinearLayout



ConstraintLayout

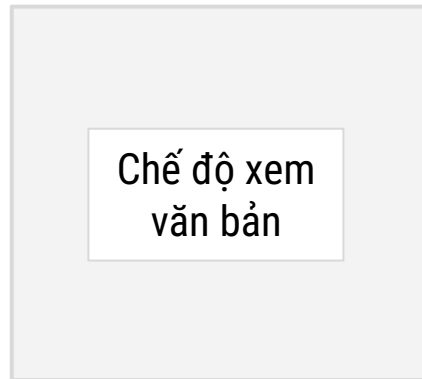


ViewGroup là chế độ xem mẹ và các chế độ xem bên trong chế độ xem đó đều là chế độ xem con.

# Ví dụ về FrameLayout

FrameLayout thường lưu giữ một Chế độ xem con.

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Hello World!"/>
</FrameLayout>
```





# Ví dụ về LinearLayout

- Căn chỉnh các chế độ xem con trong một hàng hoặc cột
- Đặt `android:orientation` thành ngang hoặc dọc

## <LinearLayout

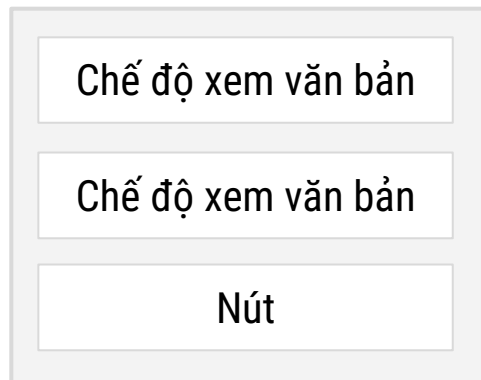
```
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical">
```

```
    <TextView ... />
```

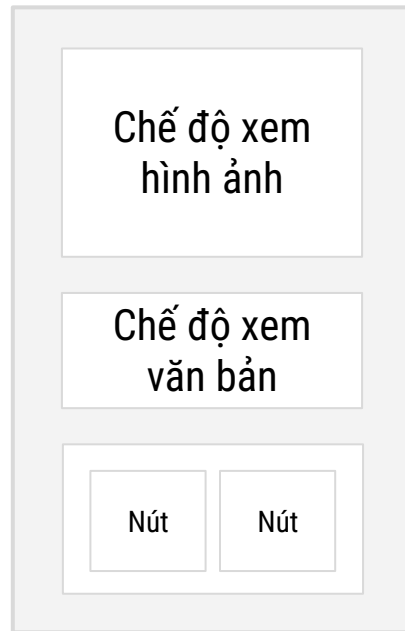
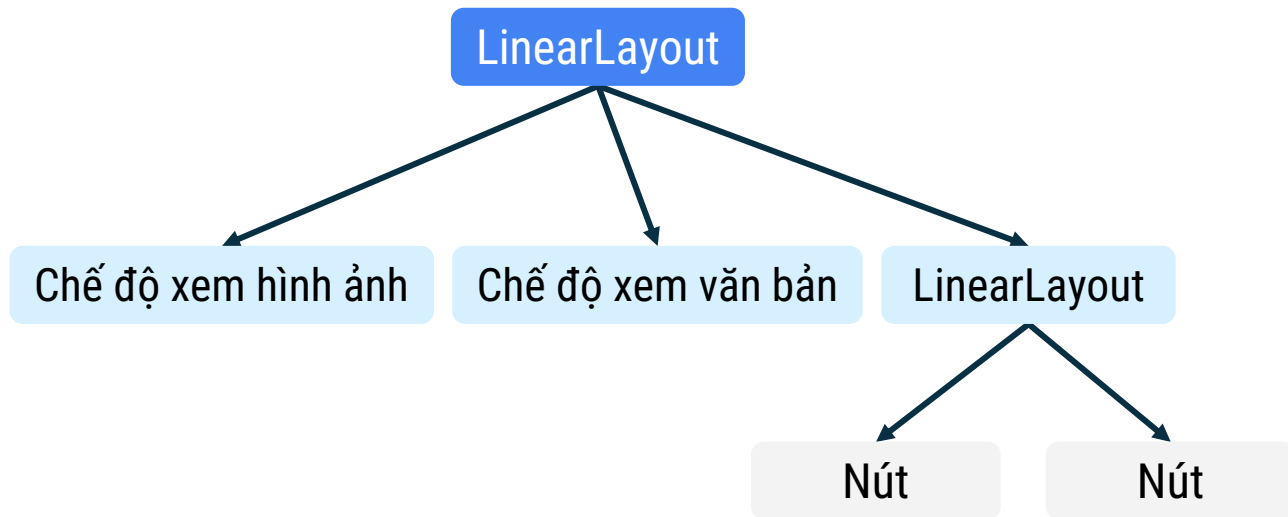
```
    <TextView ... />
```

```
    <Button ... />
```

## </LinearLayout>



# Hệ phân cấp chế độ xem



# Tài nguyên ứng dụng

Nội dung tĩnh hoặc các tệp bổ sung mà mã của bạn sử dụng

- Tệp bố cục
- Hình ảnh
- Tệp âm thanh
- Chuỗi giao diện người dùng
- Biểu tượng ứng dụng

# Thư mục tài nguyên phổ biến

Thêm các tài nguyên vào ứng dụng của bạn bằng cách đưa vào thư mục tài nguyên thích hợp trong thư mục `res` mẹ.

```
main
├── java
├── res
│   ├── drawable
│   ├── layout
│   ├── mipmap
│   └── values
```

# Mã tài nguyên

- Mỗi tài nguyên đều có một mã tài nguyên dùng để truy cập.
- Khi đặt tên cho tài nguyên, bạn cần tuân theo quy ước là dùng toàn bộ chữ thường có dấu gạch dưới (ví dụ: `activity_main.xml`).
- Android sẽ tự động tạo một tệp lớp có tên là `R.java` với thông tin tham chiếu đến mọi tài nguyên trong ứng dụng.
- Từng mục được tham chiếu bằng:

`R.<resource_type>.<resource_name>`

Ví dụ: `R.drawable.ic_launcher` (`res/drawable/ic_launcher.xml`)  
`R.layout.activity_main` (`res/layout/activity_main.xml`)

# Mã tài nguyên cho các chế độ xem

Từng chế độ xem cũng có thể có mã tài nguyên.

Thêm thuộc tính `android:id` vào Chế độ xem trong tệp XML. Dùng cú pháp `@+id/name`.

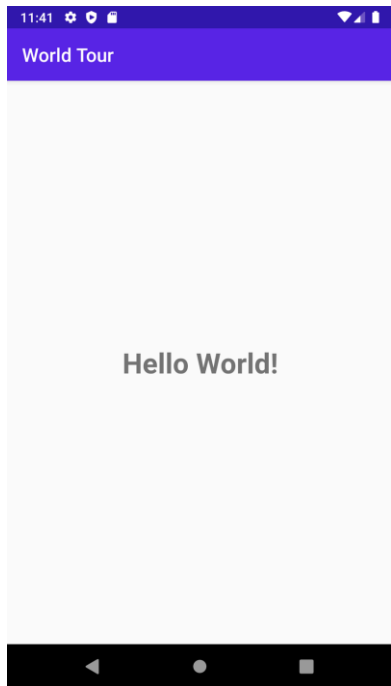
```
<TextView  
    android:id="@+id/helloTextView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World!"/>
```

Trong ứng dụng của mình, bạn hiện có thể tham chiếu đến Chế độ xem văn bản cụ thể này bằng:

```
R.id.helloTextView
```

# Hoạt động

# Hoạt động là gì?



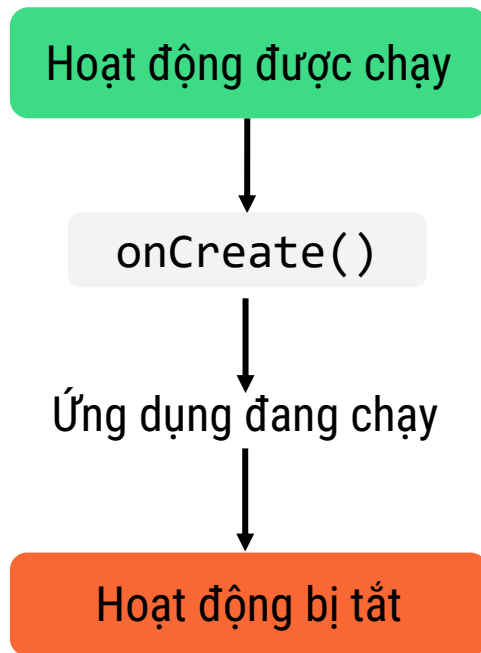
- Hoạt động là phương tiện để người dùng hoàn thành một mục tiêu chính.
- Một ứng dụng Android bao gồm một hoặc nhiều hoạt động.



# MainActivity.kt

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

# Cách thức chạy của một Hoạt động

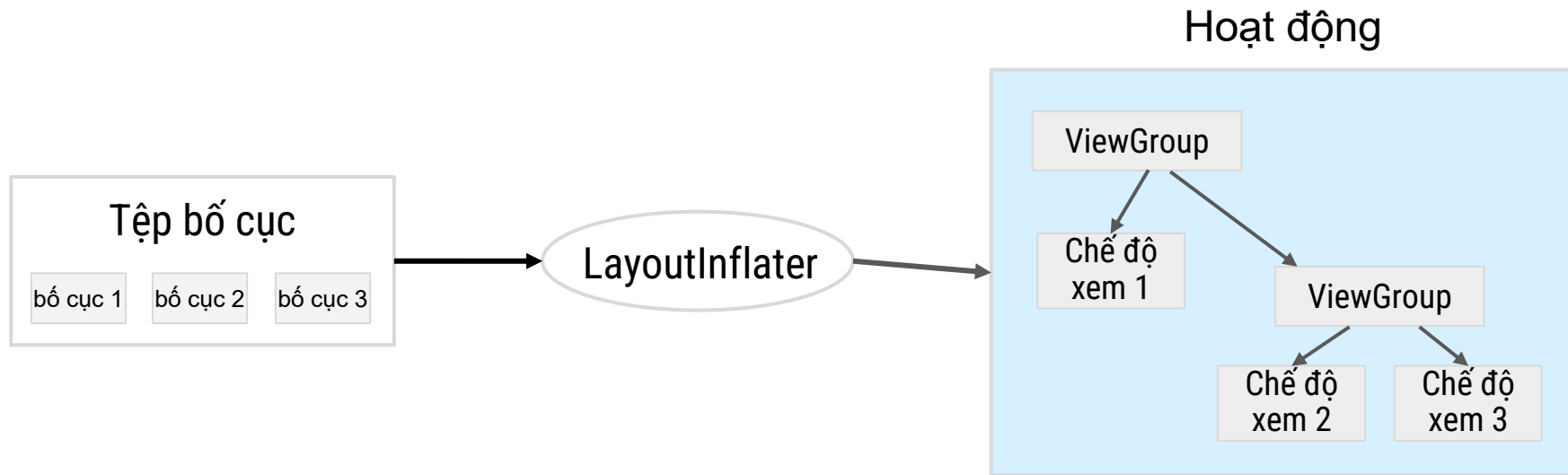


# Triển khai lệnh gọi lại onCreate()

Được gọi khi hệ thống tạo Hoạt động của bạn

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
}
```

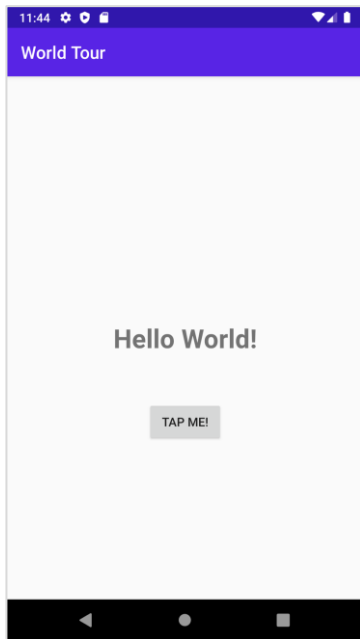
# Tăng cường bố cục



# Tạo một ứng dụng giàu tính tương tác

# Xác định hành vi của ứng dụng trong Hoạt động

Sửa đổi Hoạt động để ứng dụng phản hồi hoạt động đầu vào của người dùng, chẳng hạn như một lượt nhấn nút.



# Sửa đổi Chế độ xem một cách linh động

Trong tệp `MainActivity.kt`:

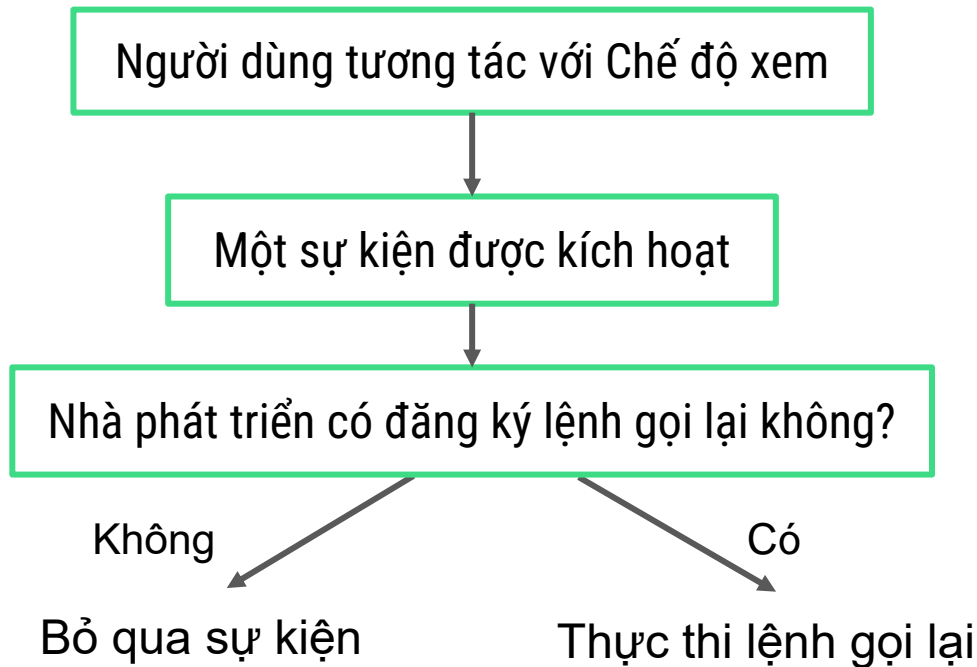
Lấy thông tin tham chiếu đến Chế độ xem trong hệ phân cấp chế độ xem:

```
val resultTextView: TextView = findViewById(R.id.textview)
```

Thay đổi các thuộc tính hoặc phương thức gọi trên thực thể Chế độ xem:

```
resultTextView.text = "Goodbye!"
```

# Thiết lập trình xử lý cho các sự kiện cụ thể





# View.OnClickListener

```
class MainActivity : AppCompatActivity(), View.OnClickListener {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
        val button: Button = findViewById(R.id.button)  
        button.setOnClickListener(this)  
    }  
  
    override fun onClick(v: View?) {  
        TODO("not implemented")  
    }  
}
```

# SAM (phương thức trừu tượng đơn)

Chuyển đổi một hàm thành phương thức triển khai giao diện

**Định dạng:** `InterfaceName { lambda body }`

```
val runnable = Runnable { println("Hi there") }
```

is equivalent to

```
val runnable = (object: Runnable {  
    override fun run() {  
        println("Hi there")  
    }  
})
```

# View.OnClickListener làm phương thức trừu tượng đơn (SAM)

Một cách ngắn gọn hơn để khai báo trình xử lý lượt nhấp

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
  
        val button: Button = findViewById(R.id.button)  
        button.setOnClickListener({ view -> /* do something*/ })  
    }  
}
```

# Khởi tạo trễ

```
class Student(val id: String) {  
    lateinit var records: HashSet<Any>  
  
    init {  
        // retrieve records given an id  
    }  
}
```

# Ví dụ về lateinit trong Hoạt động

```
class MainActivity : AppCompatActivity() {  
  
    lateinit var result: TextView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
        result = findViewById(R.id.result_text_view)  
    }  
}
```

# Gradle: Xây dựng một ứng dụng Android

# Gradle là gì?

- Xây dựng hệ thống tự động hóa
- Quản lý chu kỳ xây dựng thông qua một loạt tác vụ (ví dụ: biên dịch các nguồn Kotlin, chạy các hoạt động kiểm tra, cài đặt ứng dụng cho thiết bị)
- Xác định thứ tự thích hợp để chạy các tác vụ
- Quản lý phần phụ thuộc giữa các dự án và thư viện của bên thứ ba

# Tập bản dựng Gradle

- Khai báo trình hỗ trợ
- Xác định các thuộc tính Android
- Xử lý các phần phụ thuộc
- Kết nối với kho lưu trữ



# Trình hỗ trợ

Cung cấp các thư viện và cơ sở hạ tầng cần thiết cho ứng dụng của bạn

apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'

# Cấu hình Android

```
android {  
    compileSdkVersion 30  
    buildToolsVersion "30.0.2"  
  
    defaultConfig {  
        applicationId "com.example.sample"  
        minSdkVersion 19  
        targetSdkVersion 30  
    }  
}
```

# Phần phụ thuộc

```
dependencies {  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-  
jdk7:$kotlin_version"  
    implementation 'androidx.core:core-ktx:1.3.2'  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.material:material:1.2.1'  
    ...  
}
```

# Kho lưu trữ

```
repositories {  
    google()  
    mavenCentral()  
}
```

# Tác vụ Gradle phổ biến

- Dọn sạch
- Tác vụ
- InstallDebug

# Hỗ trợ tiếp cận

# Hỗ trợ tiếp cận

- Đề cập đến việc cải thiện thiết kế và chức năng của ứng dụng để giúp nhiều người hơn, kể cả những người khuyết tật, dễ dàng sử dụng
- Việc làm cho ứng dụng dễ tiếp cận hơn sẽ mang lại trải nghiệm chung tốt hơn cho người dùng và giúp tất cả người dùng đều được hưởng lợi

# Làm cho ứng dụng dễ tiếp cận hơn

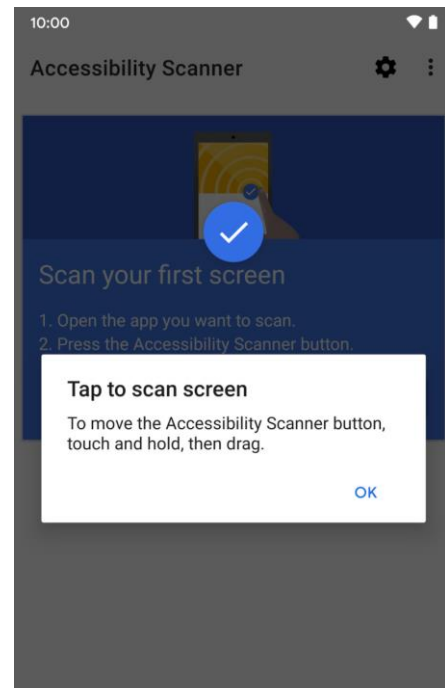
- Tăng khả năng nhìn thấy văn bản với tỷ lệ tương phản màu giữa nền trước và nền sau:
  - Tối thiểu là 4,5:1 cho văn bản nhỏ so với nền
  - Tối thiểu là 3:1 cho văn bản lớn so với nền
- Dùng các thành phần điều khiển lớn, đơn giản
  - Kích thước đích chạm tối thiểu phải là 48dp x 48dp
- Mô tả từng thành phần trên giao diện người dùng
  - Đặt phần mô tả nội dung trên các hình ảnh và thành phần điều khiển



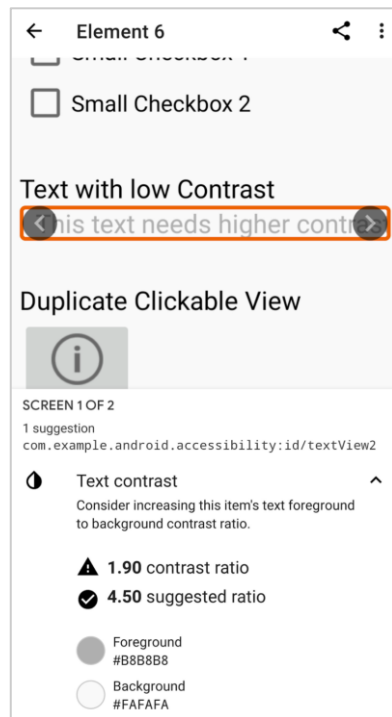
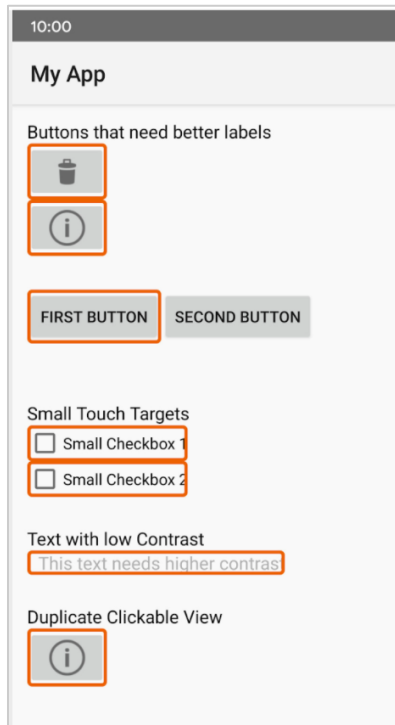
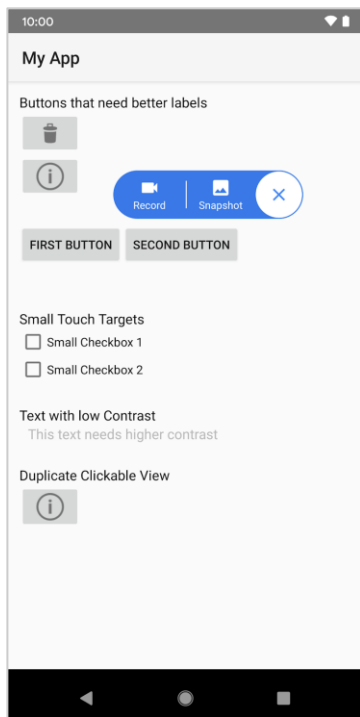
# Trình quét hỗ trợ tiếp cận

Công cụ quét màn hình của bạn và đề xuất những điểm cải tiến để làm cho ứng dụng dễ tiếp cận hơn, dựa trên:

- Nhãn nội dung
- Kích thước đích chạm
- Chế độ xem có thể nhấp
- Độ tương phản của văn bản và hình ảnh



# Ví dụ về Trình quét hỗ trợ tiếp cận



# Thêm nhãn nội dung

- Đặt thuộc tính `contentDescription` → đọc to bằng trình đọc màn hình

```
<ImageView  
    ...  
    android:contentDescription="@string/stop_sign" />
```

- Văn bản trong Chế độ xem văn bản đã được cung cấp cho các dịch vụ hỗ trợ tiếp cận, không cần có nhãn bổ sung

# Không cần có nhãn nội dung

- Đối với các thành phần đồ họa chỉ dùng cho mục đích trang trí, bạn có thể đặt

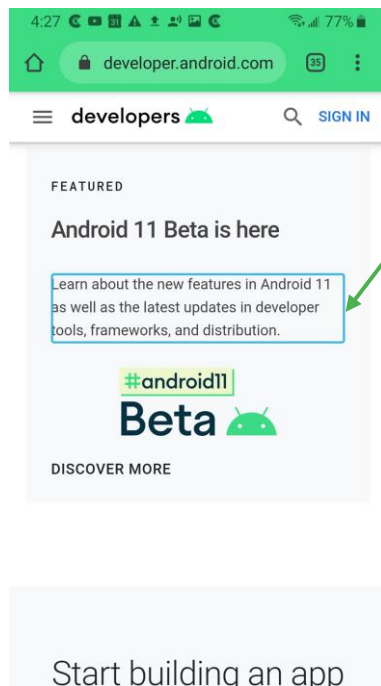
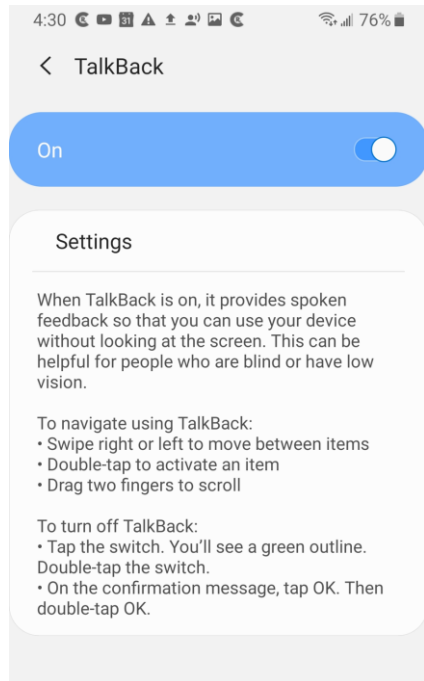
```
android:importantForAccessibility="no"
```

- Tốt hơn là bạn nên xóa các thông báo không cần thiết cho người dùng

# TalkBack

- Trình đọc màn hình của Google đi kèm với các thiết bị Android
- Cung cấp phản hồi bằng giọng nói để bạn không phải nhìn vào màn hình khi sử dụng thiết bị
- Cho phép bạn thao tác trên thiết bị bằng cử chỉ
- Có bàn phím chữ nổi cho chữ nổi tiếng Anh hợp nhất

# Ví dụ về TalkBack



Đọc to văn bản khi người dùng di chuyển trên màn hình

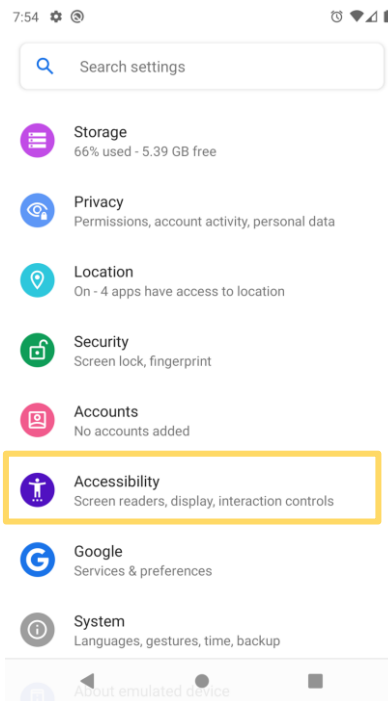
# Tiếp cận bằng công tắc

- Cho phép điều khiển thiết bị bằng một hoặc nhiều công tắc thay vì màn hình cảm ứng
- Quét giao diện người dùng ứng dụng của bạn và làm nổi bật từng mục cho đến khi bạn lựa chọn
- Dùng với công tắc bên ngoài, bàn phím ngoài hoặc các nút trên thiết bị Android (ví dụ: nút âm lượng)

# Bộ hỗ trợ tiếp cận của Android

Tập hợp các ứng dụng hỗ trợ tiếp cận giúp bạn sử dụng thiết bị Android mà không cần nhìn vào thiết bị hoặc dùng một thiết bị công tắc. Bộ ứng dụng này bao gồm:

- Trình đọc màn hình TalkBack
- Tiếp cận bằng công tắc
- Trình đơn hỗ trợ tiếp cận
- Chọn để nói





# Tài nguyên về hỗ trợ tiếp cận

- [Xây dựng các ứng dụng dễ tiếp cận hơn](#)
- [Các nguyên tắc để cải thiện tính năng hỗ trợ tiếp cận của ứng dụng](#)
- [Lớp học lập trình cơ bản về Hỗ trợ tiếp cận trên Android](#)
- [Các phương pháp hay nhất về Material Design liên quan đến hỗ trợ tiếp cận](#)

# Tóm tắt

# Tóm tắt

Trong Bài học 4, bạn đã tìm hiểu cách:

- Dùng Chế độ xem và `ViewGroup` để xây dựng giao diện người dùng của ứng dụng
- Truy cập vào các tài nguyên trong ứng dụng từ `R.<resource_type>.<resource_name>`
- Xác định hành vi của ứng dụng trong Hoạt động (ví dụ: đăng ký `OnClickListener`)
- Dùng Gradle làm hệ thống xây dựng để xây dựng ứng dụng
- Thực hiện theo các phương pháp hay nhất để làm cho ứng dụng dễ tiếp cận hơn

# Tìm hiểu thêm

- [Bố cục](#)
- [LinearLayout](#)
- [Tổng quan về sự kiện đầu vào](#)
- [Chế độ xem](#)
- [ViewGroup](#)



# Lộ trình

Thực hành những gì bạn đã học được bằng cách hoàn thành lộ trình này:

[Bài học 4: Xây dựng ứng dụng Android đầu tiên](#)

