ANDROID

SESSION FOUR

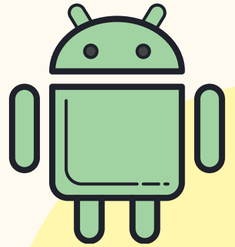
Agenda

- Intro To Android Development
- Anatomy of Android App Project
- Activity and Activity life cycle
- Logging Statement
- Intro to XML
- View and View sub Classes
- Layouts

INTRO TO ANDROID

Intro to Android

Think about your smartphone. Chances are, it's running Android. This versatile operating system gives you access to millions of apps, lets you personalize your device to fit your style, and connects you to the world. From social media to games, productivity tools to entertainment, Android puts it all at your fingertips. We'll look at how Android makes our mobile experiences richer and more convenient.



What is Android

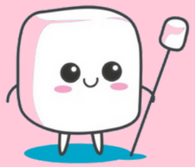
- Open-source mobile operating system developed by Google.
- Used on smartphones, tablets, and other smart devices.
- Supports multiple programming languages, including Java, Kotlin
- Security and Updates
- Android versions :

Cupcake 1.5

Donut 1.6

Lollipop 5.0

Marshmallow 6.0



Nougat 7.0

Android 10

android

Android 13

Official IDE

Android Studio

- Easy for running and Debugging :provides powerful debugging tools like Logcat.
- Supports Unit, UI, and Integrated Testing
- create virtual devices (Emulators)
- Visual Layout Editor :A drag-and-drop UI editor that simplifies designing
- Project View & Android View :
  - Project View: Displays the complete file structure, including Gradle scripts and resources.
  - Android View: Organizes files in a developer-friendly way, focusing on Java/Kotlin, res, and manifest files.

Android Main Components

Android apps are built using ***Four core components*** that define their structure and behavior:

- Activity
  - Represent a single screen in an app.
  - One main Task
  - Example: A LoginActivity for user authentication.
- Service
  - Faceless Task runs in The background
  - Used for long-running tasks like playing music
  - Example: Download App in The Background.
- Broadcast Receiver
  - Used To respond to The notification or status change
- Content Provider
  - Enables the apps To share The data

Let’s Take a tour in Android Studio

Android App Project

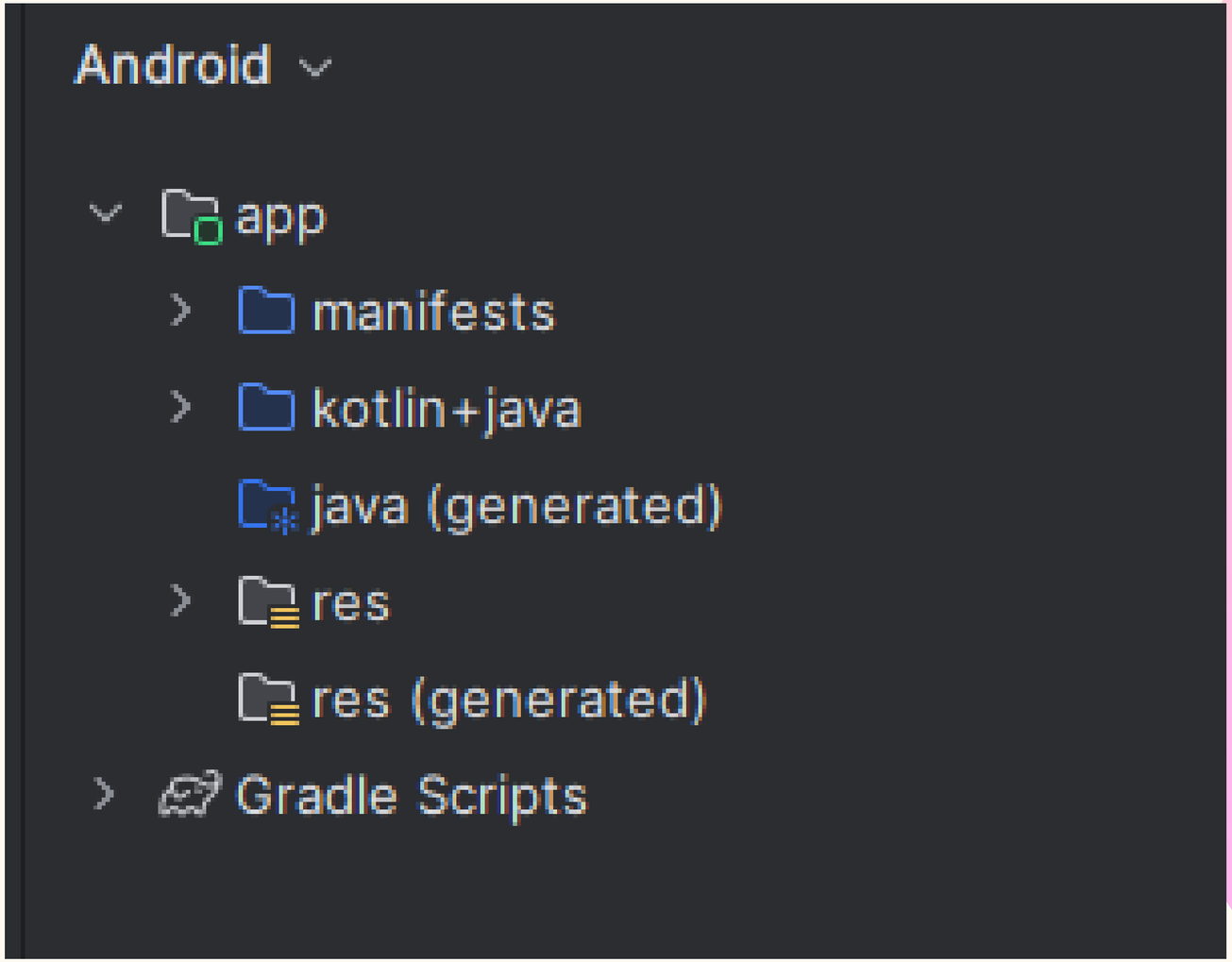
What is the Anatomy of an Android App?

- The structure and organization of files that make up an Android project.
- Helps developers manage code, resources, and configurations efficiently.
- Each file has a specific role in app development.

Key Components of an Android Project

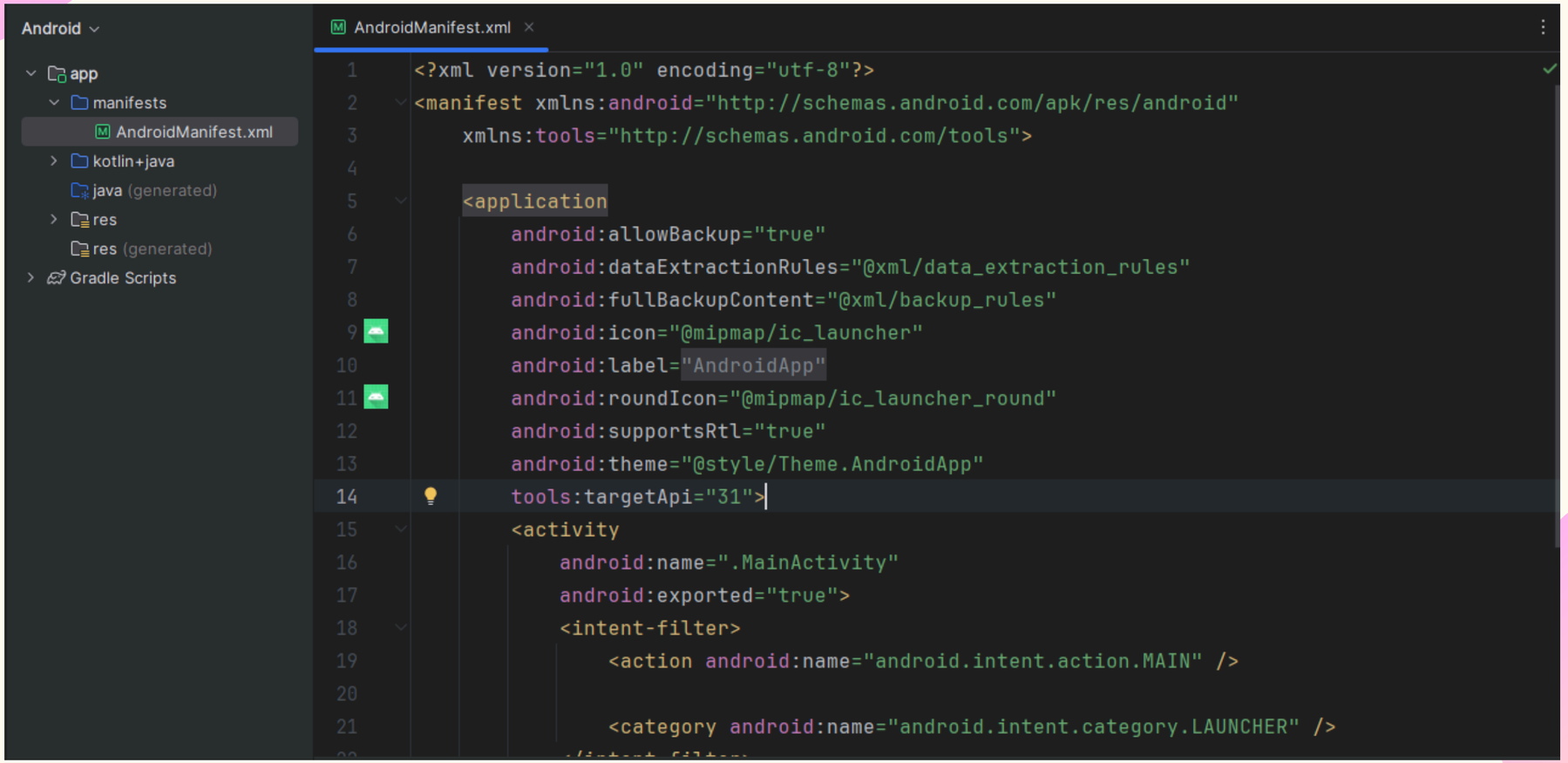
- AndroidManifest.xml
- java/ or kotlin/ folder
- res/
- Gradle Files

Project Structure Overview



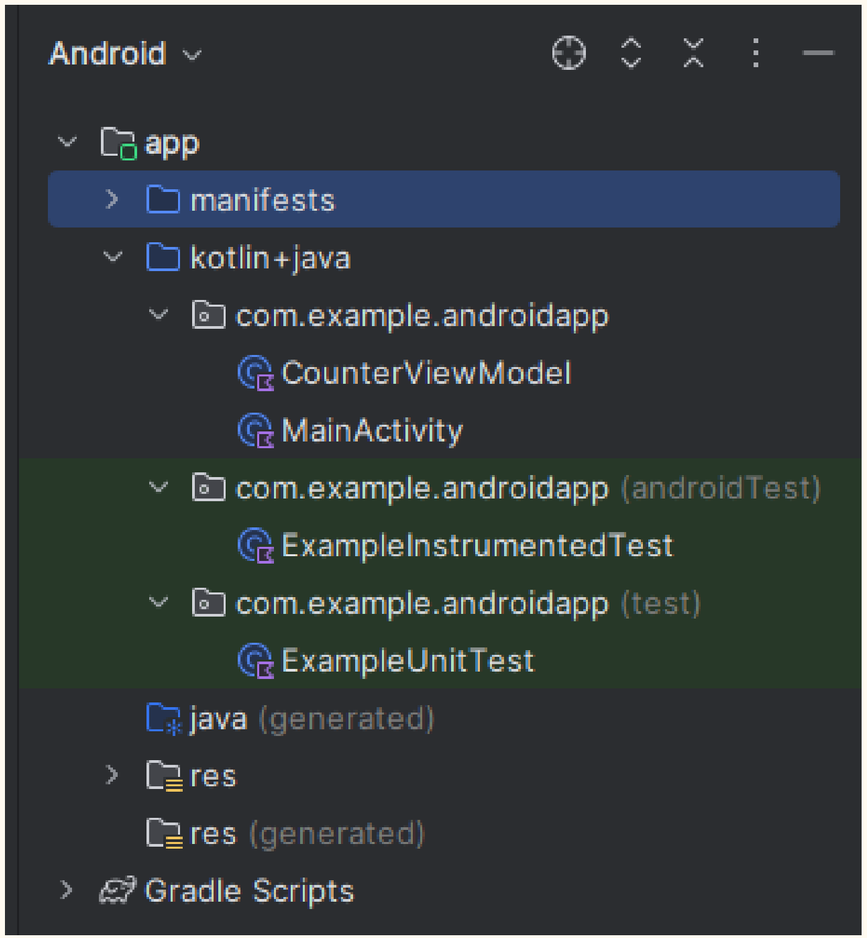
AndroidManifest.xml

- Essential configuration file for every Android app.
- Declares permissions, activities, services, broadcast receivers, and hardware features.



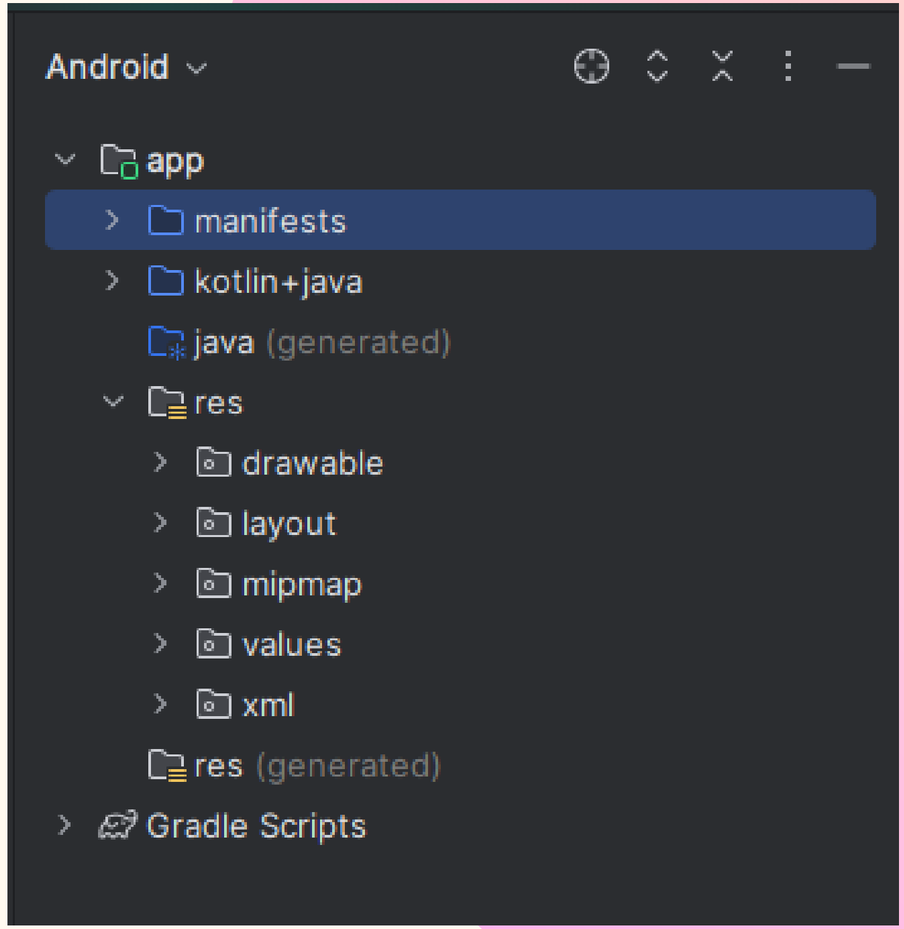
Kotlin+java Folder

- Contains all the app’s source code.
- Organized into packages for better structure and modularity.



res/ Folder (Resources)

- Holds non-code UI elements.
- Subfolders:
  - **layout/** - XML files defining UI structure.
  - **drawable/** - Images, shapes, and icons.
  - **values/** - Strings, colors, and dimensions
  - **mipmap/** - App Icon(Logo)
  - **raw/** - For The Animation , Audio Files (Json , Text files)

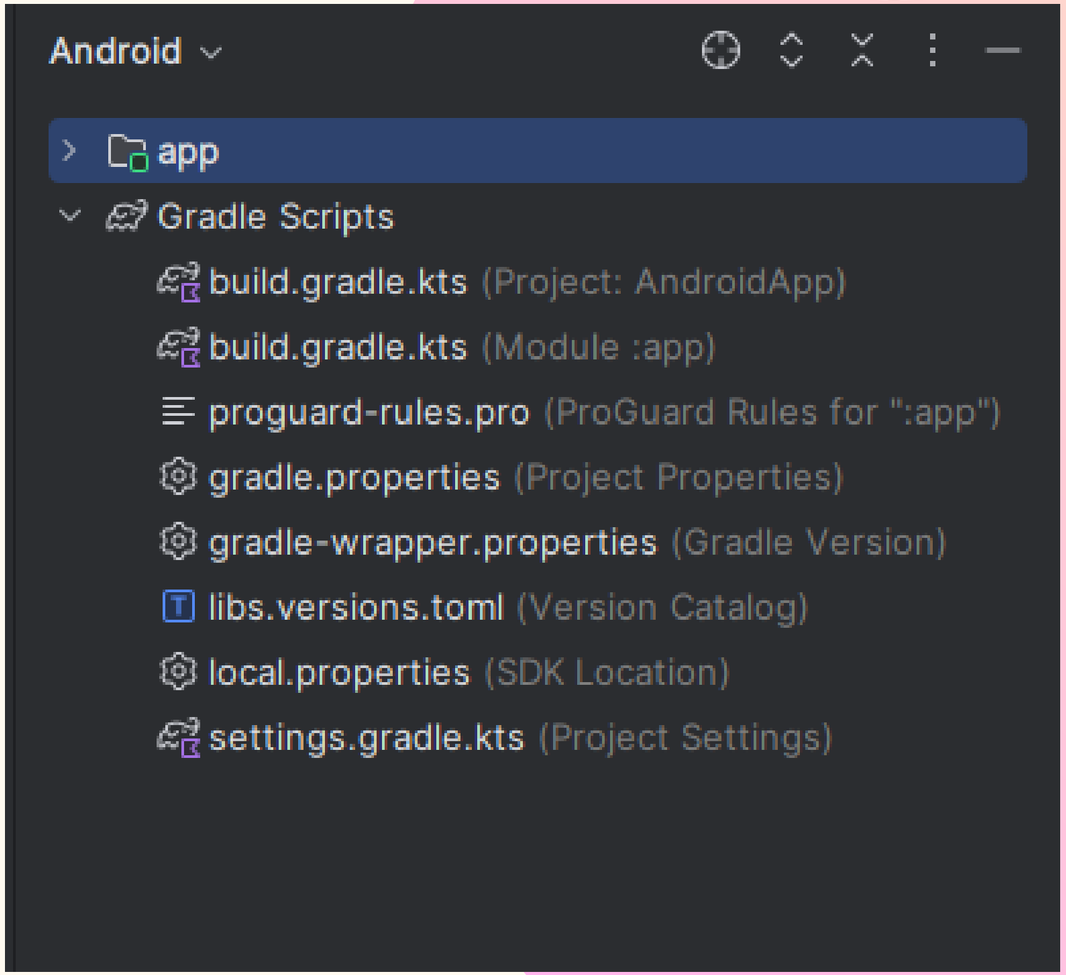


Android Releases and API Levels

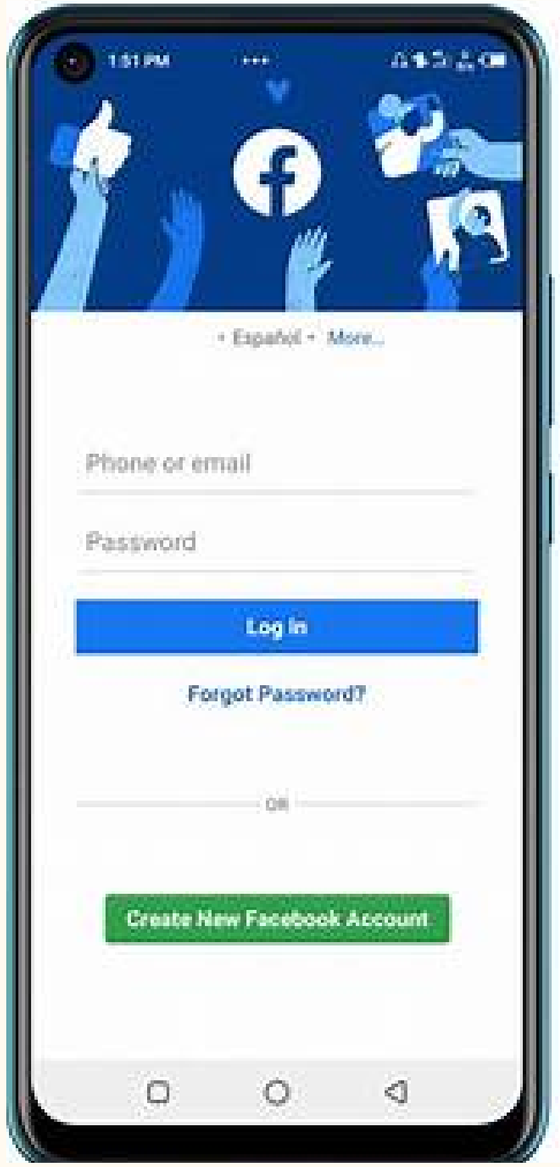
Platform Version	API Level	VERSION_CODE
Android 10.0	29	Q
Android 9	28	P
Android 8.1	27	O_MR1
Android 8.0	26	O
Android 7.1.1 Android 7.1	25	N_MR1
Android 7.0	24	N
Android 6.0	23	M
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP

Gradle Build System

- What is Gradle?
  - Manages dependencies and build configurations.
  - Automates the build process.
- **build.gradle (Module-level):** Defines dependencies.
- In **build.gradle** (mod -level) :
  - minSdk <= targetSdk <= compileSdk



Activity


1:51 PM
• Español • More...
Phone or email
Password
Log In
Forgot Password?
OR
Create New Facebook Account

Activity

What is an Activity?

- An Activity represents a single screen with a user interface.
- Handling user interactions
- Each Activity is independent but can communicate with other activities through intents.
- The App Can have Zero or more activity
- Once The activity is started , pushed in the back stack and take the users focus

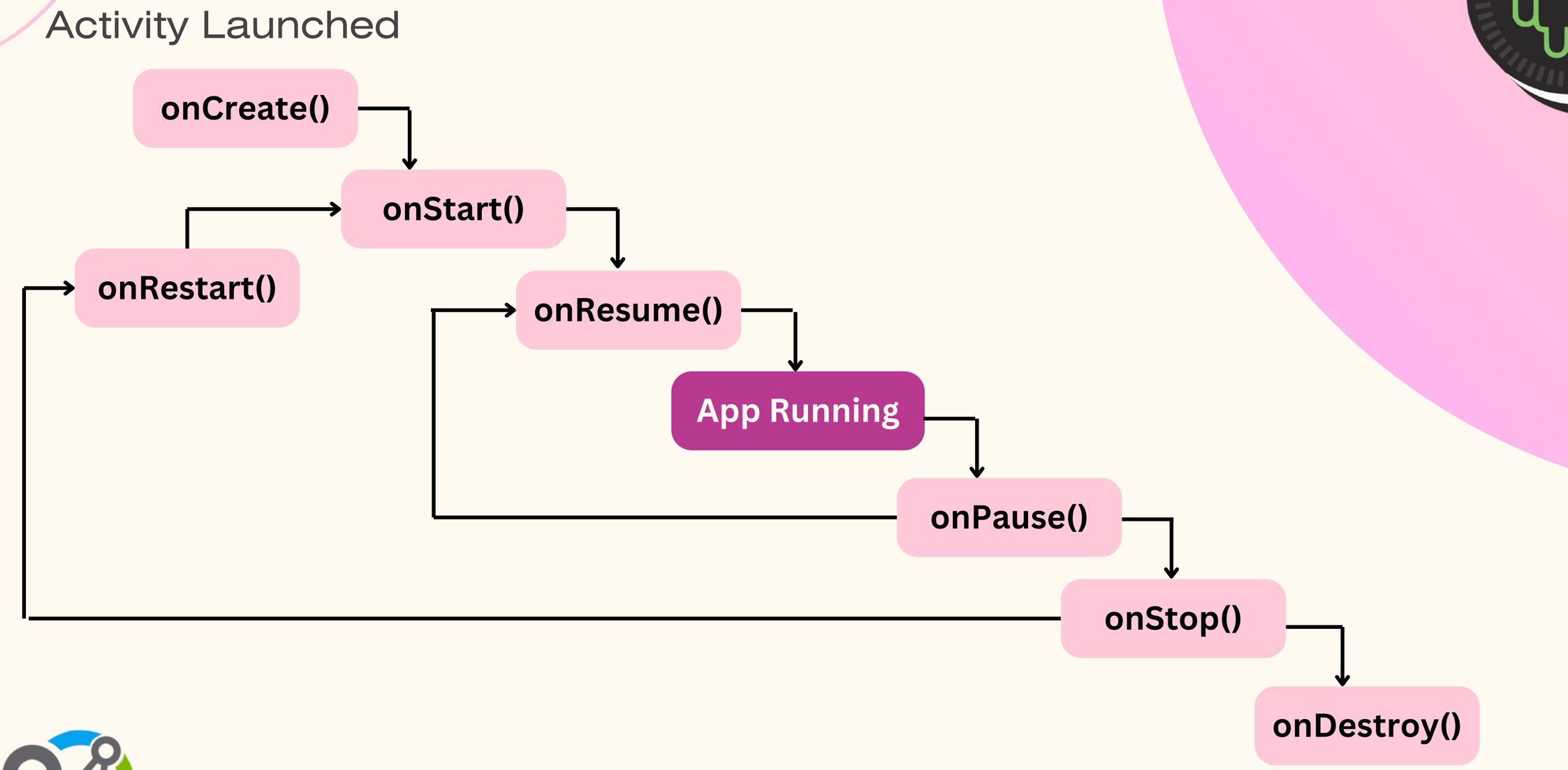
Activity

How To Create Activity :

1. Create XML file in layout folder.
2. Create A Class Extend Activity or One Of It’s Subclasses
    - class MainActivity : AppCompatActivity()
3. Connect The Activity With The layout using setContentView()
    - setContentView(R.layout.activity_main)
4. Declare The Activity In Android Manifest
    - using <activity> tag

Let’s See in Android Studio These Steps

Activity Lifecycle



Activity Lifecycle

- **onCreate():** Called when the Activity is first created; initializes UI and data.
- **onStart():** Activity started to be visible to the user but is not yet interactive.
- **onResume():** The Activity is now in the foreground and ready to interact with the user.
- **onPause():** Activity is partially obscured (e.g., another Activity opens in front of it).
- **onStop():** Activity is completely hidden but still exists in memory.
- **onDestroy():** Activity is terminated and removed from memory.

Activity Lifecycle

- **onCreate()**
  1. The first method get Called when the activity first created
  2. Initialize your activity. This is where you should set up your UI, initialize data that the activity will need
  3. Inflate the activity UI, Startup other code logic
- **onStart()**
  1. Activity ready to be visible, but not yet interactive.
  2. you can start any media player , video , animation.

Activity Lifecycle

- **onResume()**

  Activity gains Input Focus (User Can Interact with).

- **onPause()**

  1. Activity has lost Focus(not in the fore ground).
  2. Activity is still visible(partially) , but user is n’t interacting with it.
  3. Stop animations, sound ...

- **onStop()**

  1. Activity is no longer visible .
  2. release the resources that’s not needed any more
  3. save any persistent state / data.

Activity Lifecycle

- **onDestroy()**

  Activity is about to be destroyed :

  - The activity finished or the configuration has been changed.
  - perform any clean up of the resources

- **onRestart()**

  1. Reinitializing UI elements or data that was paused or removed in onStop().
  2. Restore resources that were released

Activity Lifecycle

- **onSaveInstanceState()**
  1. t allows you to save the current state of the activity, such as the user's input, UI elements' states during configuration changes.
  2. saved in a **Bundle.**

- **onRestoreInstanceState ()**
  1. Is called after the activity is recreated (after a configuration change)
  2. Configuration Changes :
     a. Rotates The Device
     b. Choose Different System Language (local in the App)
     c. multi-window mode.
     d. Dark Mode

Logging Statement

What is Logging?

- Logging is the process of tracking events that happen in an application.
- Helps developers debug, monitor performance, and understand application behavior.

```
Log.d("TAG", "Debug message")
```

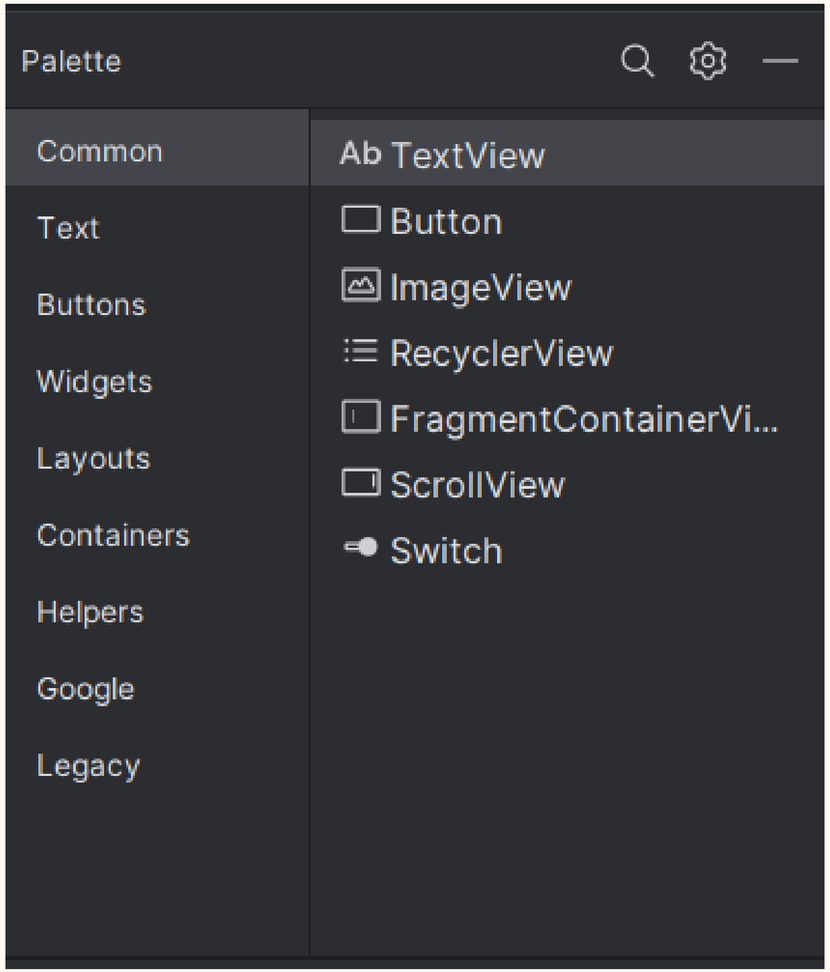
Let’s See The Life Cycle implementation

INTRO TO XML

Visual Layout Editor

1- Palette

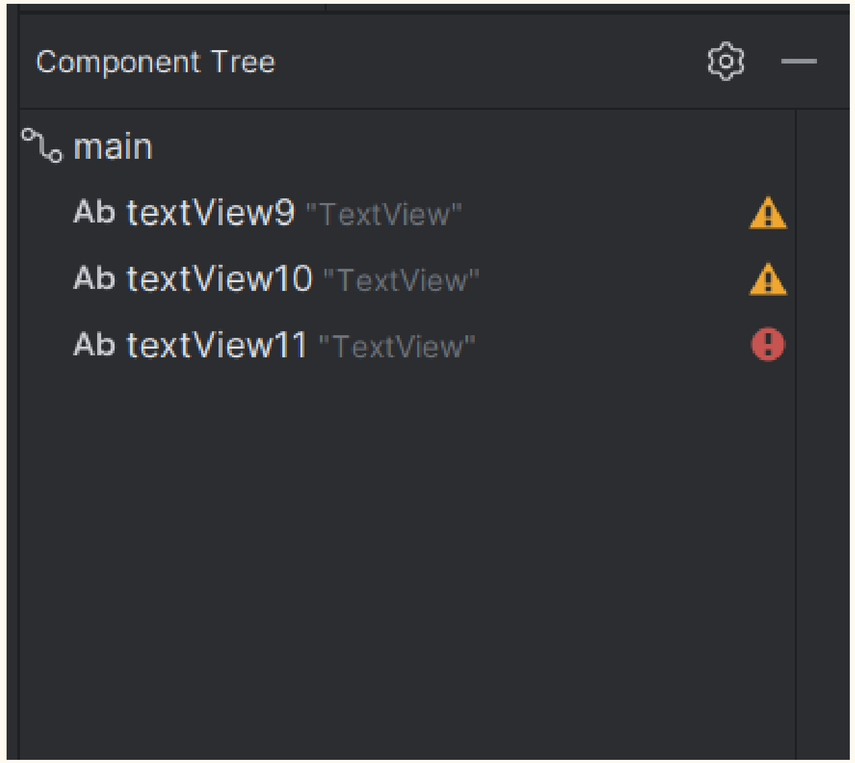
Contains Buttons, TextViews, ImageViews, RecyclerViews, and other UI components that you can drag into your layout.



Visual Layout Editor

2-Component Tree

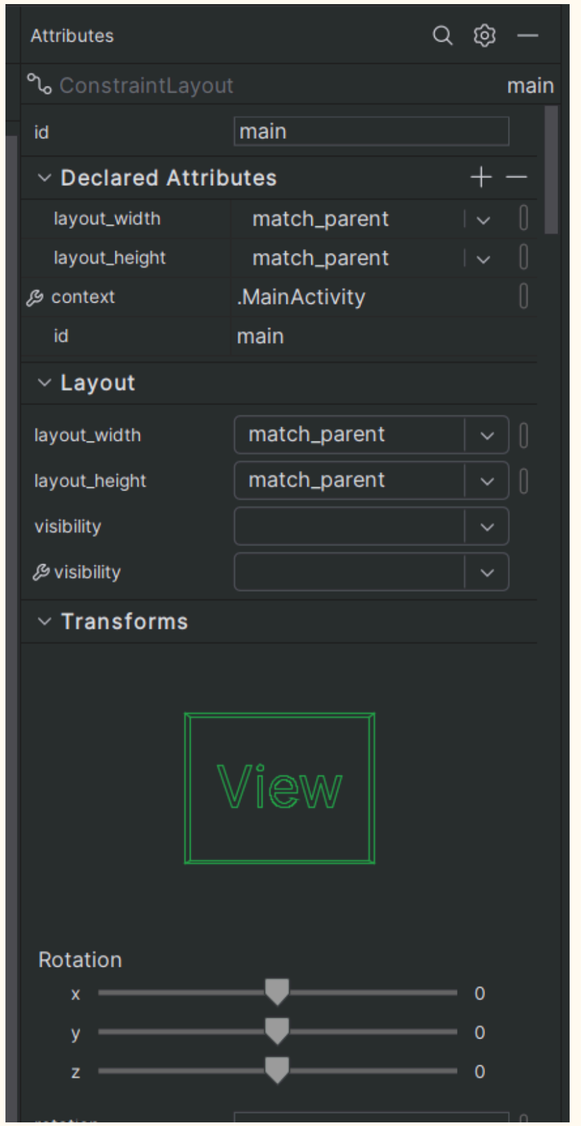
Displays a hierarchical structure of the layout, showing parent-child relationships between UI elements.



Visual Layout Editor

3-Attribute Panel

Allows you to customize width, height, text size, padding, margins, constraints, colors and all other attributes.



XML

XML (Extensible Markup Language) is a structured format used in Android development for defining UI layouts, configuration files

Key Features of XML

- Self-descriptive: Uses tags to define elements clearly.
- Hierarchical structure: Data is nested within elements.
- Lightweight & readable: Easy to understand and edit.

Why Use XML in Android?

- Separates UI from logic → UI is in **XML**, logic in **Kotlin+Java.**
- Easy to read & modify → Changes in UI don’t require code changes.
- Declare the UI in XML→ makes it easier to visualize the structure.

VIEW AND VIEW SUBCLASSES

View and View Subclasses

- Every UI element you see on the screen (buttons, text fields, images, etc.) is a View or a subclass of View . In Android, View is the base class for all UI components

- **View :** is The Base Class for all visual interfaces.
  - Commonly Known as Controls and Widgets
  - All UI Components including the layout Classes are Derived from **View** Class
  - Examples : TextView , Button, ScrollView , ImageView, **ViewGroups**.

View and View Subclasses

- **Views :** User Interface Building blocks in Android
  - Bounded by a rectangle area in the screen
  - Can be grouped to form more **Complex interfaces.**
  - Examples : TextView , Button , ImageView

Let’s See The Difference...

View and View Subclasses

View Groups :

A ViewGroup is a container that holds **multiple Views** (or other ViewGroups) and manages their layout on the screen.

ViewGroup = A special type of View that holds **other Views**
View = A single UI component (Button, TextView, etc.)

Examples : LinearLayout, ConstraintLayout, RelativeLayout

Let’s See More Details ..

LAYOUTS

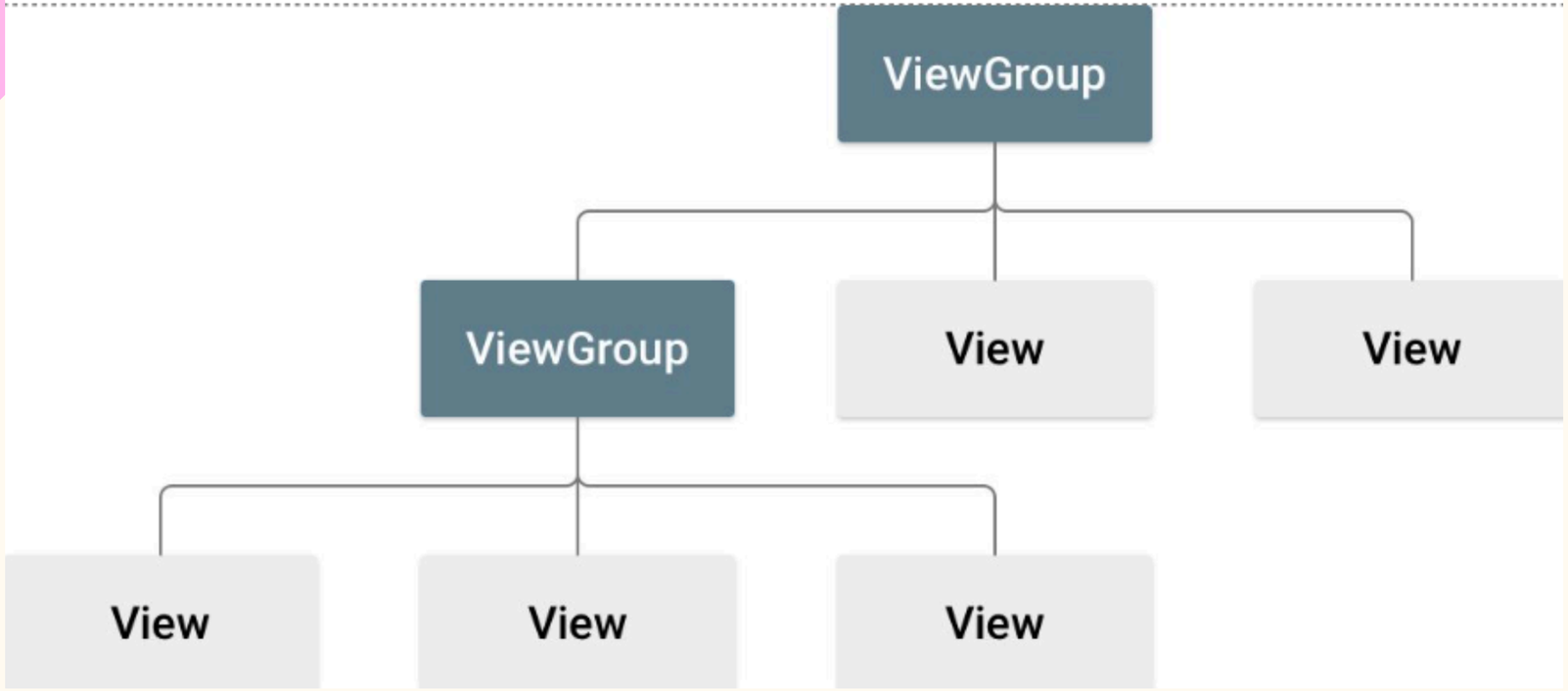
Layouts

Layouts in Android define how UI elements (Views) are arranged on the screen. Each layout type has different rules for positioning and organizing Views.

- Layouts is a **Subclasses** of the ***View Group.***
- A container that arranges UI elements (Views).
- Can be in Row , Column , Grid, Table.
- ViewGroup Objects is The **branches** of the Tree
- Other View Objects are the **leaves** of the Tree

Layouts

The Root is always a View Group



Let’s See The Hierarchy...

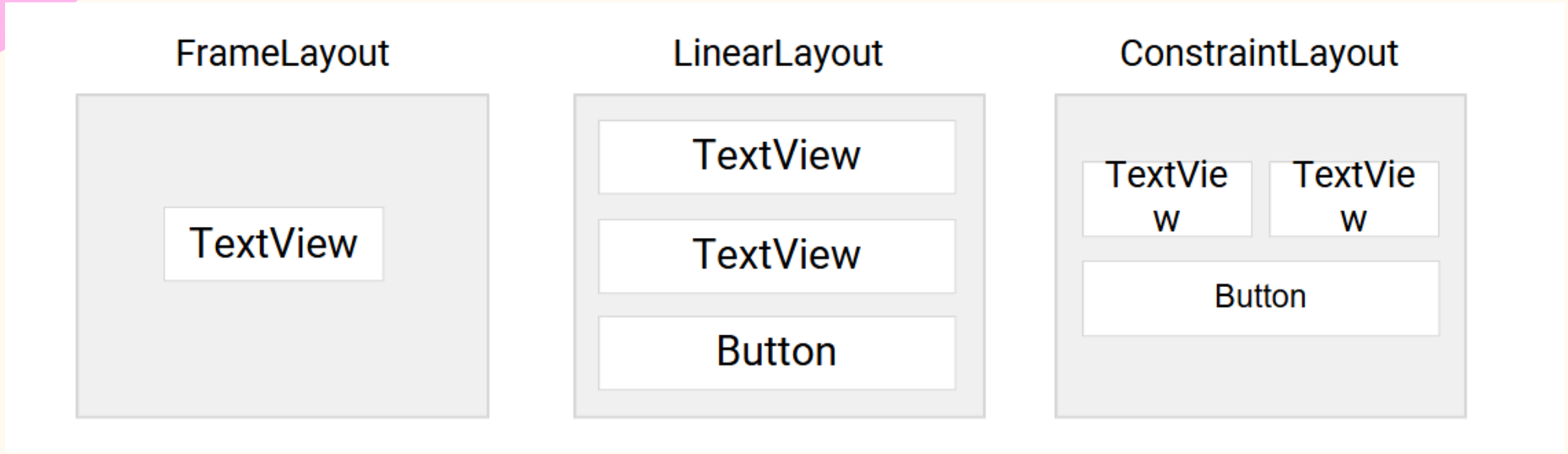
Layouts

Types of Layouts :

- **FrameLayout :** The Simplest type of Layout , Stacks Views on top of each other.
- **ConstraintLayout** : Connect The Views with constraints.
- **LinearLayout** : Horizontal or Vertical row
- **TableLayout**: Rows and Columns.
- **RelativeLayout**: child Views Relative To each Other.
- **GridLayout** :Organizes Views in rows and columns like a grid.
- **ScrollView :** Enables scrolling when content is larger than the screen.

Layouts

- **CoordinatorLayout :** Provides advanced scrolling behavior and interaction between views.



Thank You