Android Interview Questions

What is an inline function in Kotlin?
An inline function in Kotlin is a function whose bytecode is inlined at the call site, reducing overhead.

What is the advantage of using const in Kotlin?
Using const in Kotlin declares a compile-time constant, improving performance and reducing memory usage.

What is a reified keyword in Kotlin?
The reified keyword in Kotlin allows type parameters of inline functions to be accessed at runtime.

Suspending vs Blocking in Kotlin Coroutines
Suspending in Kotlin Coroutines pauses the coroutine without blocking the thread, while blocking halts the thread's execution.

Launch vs Async in Kotlin Coroutines
launch starts a coroutine that doesn't return a result, whereas async starts a coroutine that returns a Deferred result.

internal visibility modifier in Kotlin
The internal visibility modifier makes a member visible within the same module.

open keyword in Kotlin
The open keyword in Kotlin allows a class or member to be subclassed or overridden.

lateinit vs lazy in Kotlin
lateinit is used for late initialization of non-nullable properties, while lazy is used for lazy initialization of properties.

What is Multidex in Android?
Multidex in Android allows apps to include more than 65,536 methods by splitting the app into multiple DEX files.

How does the Android Push Notification system work?
The Android Push Notification system uses Firebase Cloud Messaging (FCM) to deliver messages to devices.

How does the Kotlin Multi Platform work?
Kotlin Multi Platform allows sharing common code across multiple platforms like Android, iOS, and web.

What is a ViewModel and how is it useful?
A ViewModel in Android holds and manages UI-related data, surviving configuration changes.

Is it possible to force Garbage Collection in Android?
It is not recommended to force Garbage Collection in Android, but you can suggest it using System.gc().

What is a JvmStatic Annotation in Kotlin?
The JvmStatic annotation in Kotlin generates a static method for a function in a companion object.

init block in Kotlin
The init block in Kotlin is used to initialize the class when an instance is created.

JvmField Annotation in Kotlin
The JvmField annotation in Kotlin exposes a Kotlin property as a public Java field.

singleTask launchMode in Android
The singleTask launchMode in Android ensures a single instance of the activity is created, launching it in a new task.

Difference between == and === in Kotlin
== checks for structural equality, whereas === checks for referential equality.

JvmOverloads Annotation in Kotlin
The JvmOverloads annotation generates overloaded methods for a function with default parameters.

Why is it recommended to use only the default constructor to create a Fragment?
Using only the default constructor to create a Fragment ensures proper recreation during configuration changes.

Why do we need to call setContentView() in onCreate() of Activity class?
We call setContentView() in onCreate() to define the layout resource for the activity's UI.

When only onDestroy is called for an activity without onPause() and onStop()?
onDestroy is called directly when an activity finishes due to a system constraint or crash.

What is the advantage of using const in Kotlin?
Using const in Kotlin defines compile-time constants, improving performance and reducing memory usage.

When to use lateinit keyword in Kotlin?
Use lateinit for non-nullable properties that are initialized later than their declaration.

What is an inline function in Kotlin?
An inline function in Kotlin reduces the overhead of function calls by inserting the function's code at the call site.

What are companion objects in Kotlin?
Companion objects allow defining members that belong to a class rather than to instances, simulating static members.

Remove duplicates from an array in Kotlin
Use array.distinct() to remove duplicates from an array in Kotlin.

What is a JvmStatic Annotation in Kotlin?
The JvmStatic annotation generates a static method for a function in a companion object.

What is a JvmField Annotation in Kotlin?
The JvmField annotation exposes a Kotlin property as a public Java field.

What is a JvmOverloads Annotation in Kotlin?
The JvmOverloads annotation generates overloaded methods for a function with default parameters.

noinline in Kotlin
The noinline keyword prevents a lambda parameter from being inlined in an inline function.

crossinline in Kotlin
The crossinline keyword ensures that a lambda parameter cannot use non-local returns.

Scope functions in Kotlin
Scope functions (let, run, with, also, apply) provide concise ways to operate on objects within a limited scope.

What is a reified keyword in Kotlin?
The reified keyword allows type parameters of inline functions to be accessed at runtime.

lateinit vs lazy in Kotlin
lateinit is for non-nullable properties initialized later, while lazy is for properties initialized on first access.

What is an init block in Kotlin?
The init block in Kotlin initializes class properties during object creation.

Difference between == and === in Kotlin
== checks for structural equality, whereas === checks for referential equality.

Advantage of using const in Kotlin
const improves performance and reduces memory usage by defining compile-time constants.

What are higher-order functions in Kotlin?
Higher-order functions take functions as parameters or return them, allowing for more abstract and flexible code.

What are Lambdas in Kotlin
Lambdas are anonymous functions that can be passed as arguments to higher-order functions.

associateBy - List to Map in Kotlin
The associateBy function converts a list to a map using a specified key selector.

open keyword in Kotlin
The open keyword allows a class or member to be subclassed or overridden.

Companion object in Kotlin
Companion objects allow defining members that belong to a class rather than to instances.

internal visibility modifier in Kotlin
The internal modifier makes a member visible within the same module.

partition - filtering function in Kotlin
The partition function splits a collection into two lists based on a predicate.

Infix notation in Kotlin
Infix functions provide a more readable way to call functions with a single parameter.

How does the Kotlin Multi Platform work?
Kotlin Multi Platform allows sharing common code across multiple platforms like Android, iOS, and web.

Suspending vs Blocking in Kotlin Coroutines
Suspending pauses the coroutine without blocking the thread, while blocking halts the thread's execution.

Tell some advantages of Kotlin.
Kotlin provides null safety, conciseness, interoperability with Java, and coroutine support for asynchronous programming.

What is the difference between val and var?
val defines a read-only variable, while var defines a mutable variable.

How to check if a lateinit variable has been initialized?
Use ::variable.isInitialized to check if a lateinit variable has been initialized.

How to do lazy initialization of variables in Kotlin?
Use the lazy delegate for lazy initialization of variables.

What are the visibility modifiers in Kotlin?
Kotlin's visibility modifiers are public, private, protected, and internal.

What is the equivalent of Java static methods in Kotlin?
The equivalent of Java static methods in Kotlin is methods in a companion object.

What is a data class in Kotlin?
A data class in Kotlin is a class that automatically generates standard methods like equals(), hashCode(), and toString().

How to create a Singleton class in Kotlin?
Use the object keyword to create a Singleton class in Kotlin.

What is the difference between open and public in Kotlin?
open allows a class or member to be subclassed or overridden, while public specifies visibility.

Explain the use-case of let, run, with, also, apply in Kotlin.
These scope functions provide concise ways to operate on objects within a limited scope, each with slightly different behavior.

How to choose between apply and with?
Use apply to configure an object and return it, and with to run code on an object and return a result.

Difference between List and Array types in Kotlin
A List is an immutable collection of elements, while an Array is a fixed-size collection of elements.

What are Labels in Kotlin?
Labels in Kotlin are used to identify loop statements and expressions for more readable code flow control.

What are Coroutines in Kotlin?
Coroutines provide a way to write asynchronous code sequentially, making it easier to manage background tasks.

What is Coroutine Scope?
A Coroutine Scope defines the lifecycle of coroutines, determining when they start and stop.

What is Coroutine Context?
Coroutine Context holds information about a coroutine, such as its dispatcher and job.

Launch vs Async in Kotlin Coroutines
launch starts a coroutine that doesn't return a result, while async starts a coroutine that returns a Deferred result.

When to use Kotlin sealed classes?
Use sealed classes to represent restricted class hierarchies where a type can be one of a limited set of types.

Tell about the Collections in Kotlin
Kotlin collections include List, Set, and Map, providing various functionalities for handling groups of elements.

Extension functions
Extension functions allow adding new functionality to existing classes without modifying their code.

What does ?: do in Kotlin? (Elvis Operator)
The Elvis operator returns the left-hand operand if it's not null, otherwise returns the right-hand operand.

Android Basics

Why does an Android App lag?
App lag can be caused by inefficient code, memory leaks, or poor resource management.

What is Context? How is it used?
Context provides access to resources and application-specific information.

Tell all the Android application components.
Activities, Services, Broadcast Receivers, Content Providers.

What is the project structure of an Android Application?
A typical structure includes manifest, java, res, and gradle files.

What is AndroidManifest.xml?
It's a file that contains essential information about the app, such as permissions and components.

What is the Application class?
The base class that maintains global application state.

Activity and Fragment

Why is it recommended to use only the default constructor to create a Fragment?
To ensure the fragment can be properly recreated by the system.

What is Activity and its lifecycle?
An Activity is a single screen with a user interface; its lifecycle includes states like created, started, resumed, paused, stopped, and destroyed.

What is the difference between onCreate() and onStart()?
onCreate() is called when the activity is first created, while onStart() is called when the activity becomes visible.

When only onDestroy is called for an activity without onPause() and onStop()?
When the activity is finished due to a configuration change.

Why do we need to call setContentView() in onCreate() of Activity class?
To set the user interface layout for the activity.

What is onSaveInstanceState() and onRestoreInstanceState() in activity?
Methods to save and restore the activity's UI state.

What is Fragment and its lifecycle?
A Fragment is a modular section of an activity; its lifecycle includes states like attached, created, created view, started, resumed, paused, stopped, destroyed view, and detached.

What are "launchMode"?
They define the behavior of activities regarding instance creation and task management (standard, singleTop, singleTask, singleInstance).

What is the difference between a Fragment and an Activity? Explain the relationship between the two. -
An Activity is a full-screen interface, while a Fragment is a reusable portion of the interface managed within an Activity.

When should you use a Fragment rather than an Activity?
For reusable UI components or multi-pane layouts.

What is the difference between FragmentPagerAdapter vs FragmentStatePagerAdapter?
FragmentPagerAdapter retains fragment instances, while FragmentStatePagerAdapter destroys fragments to save memory.

What is the difference between adding/replacing fragment in backstack?
Adding keeps the old fragment in memory, replacing does not.

How would you communicate between two Fragments?
Use a shared ViewModel or a callback interface.

What is retained Fragment?
A fragment that is retained across activity re-creation.

What is the purpose of addToBackStack() while committing fragment transaction?
It allows the fragment transaction to be reversed on back press.

Views and ViewGroups

What is View in Android?
A UI component like a button or text field.

Difference between View.GONE and View.INVISIBLE?
GONE removes the view from the layout, INVISIBLE hides it but retains its space.

Can you create a custom view? How?
Yes, by extending the View class and overriding its methods.

What are ViewGroups and how are they different from the Views?
ViewGroups are containers for views, providing layout structure.

What is a Canvas?
A drawing surface for custom graphics.

What is a SurfaceView?
A view for rendering content on a separate thread.

Relative Layout vs LinearLayout. -
RelativeLayout arranges views relative to each other, LinearLayout arranges them in a single direction.

Tell about Constraint Layout. -
A flexible layout that allows you to position views relative to each other or the parent container.

Do you know what is the view tree? How can you optimize its depth?
The hierarchical structure of views; optimize by flattening the hierarchy.

Displaying Lists of Content

What is the difference between ListView and RecyclerView?
RecyclerView is more flexible and efficient with view recycling and layout management.

How does RecyclerView work internally?
It uses a ViewHolder pattern for efficient view recycling and binding.

RecyclerView Optimization - Scrolling Performance Improvement -
Use ViewHolders, DiffUtil, and avoid nested layouts.

Optimizing Nested RecyclerView -
Use efficient layout managers and avoid deeply nested structures.

What is SnapHelper?
A utility that helps snap views in a RecyclerView to specific positions.

Dialogs and Toasts

What is Dialog in Android?
A small window that prompts the user to make a decision or enter information.

What is Toast in Android?
A brief message that pops up on the screen.

What is the difference between Dialog and Dialog Fragment?
DialogFragment is a fragment that displays a dialog, integrating better with the activity lifecycle.

Intents and Broadcasting

What is Intent?
A messaging object used to request an action from another app component.

What is an Implicit Intent?
An intent that does not specify a component, allowing any app that can handle the action to respond.

What is an Explicit Intent?
An intent that specifies the target component to handle the action.

What is a BroadcastReceiver?
A component that listens for and responds to system-wide broadcast announcements.

What is a Sticky Intent?
An intent that sticks around after being broadcast, so other components can retrieve the data later.

Describe how broadcasts and intents work to pass messages around your app?
Broadcasts send system-wide messages, while intents pass data and request actions within or between apps.

What is a PendingIntent?
A wrapper around an intent that allows it to be executed by another application.

Services

What is Service?
A component for performing long-running operations in the background.

Service vs IntentService -
Service runs on the main thread; IntentService runs in a background thread.

What is a Foreground Service?
A service that continues running in the foreground, displaying a notification.

What is a JobScheduler?
A manager for scheduling jobs to run in the background.

Inter-process Communication

How can two distinct Android apps interact?
Through intents, content providers, and AIDL.

Is it possible to run an Android app in multiple processes? How?
Yes, by specifying process attributes in the manifest.

What is AIDL?
Android Interface Definition Language for communication between services in different processes.

What can you use for background processing in Android?
Services, WorkManager, AsyncTask, and Kotlin Coroutines.

What is a ContentProvider and what is it typically used for?
A component for managing shared app data.

Long-running Operations

How to run parallel tasks and get a callback when all are complete?
Using Kotlin Coroutines with async and awaitAll.

What is ANR? How can the ANR be prevented?
Application Not Responding; prevent by avoiding long operations on the main thread.

What is an AsyncTask (Deprecated in API level 30)?
A class for performing background operations and publishing results on the UI thread.

What are the problems in AsyncTask?
Memory leaks and context retention.

Explain Looper, Handler, and HandlerThread.
Looper runs a message loop for a thread; Handler sends and processes messages; HandlerThread creates a thread with a Looper.

Android Memory Leak and Garbage Collection
Memory leaks occur when objects are not properly disposed of; garbage collection reclaims memory.

Working With Multimedia Content

How do you handle bitmaps in Android as it takes too much memory?
Use BitmapFactory options for scaling and caching.

Tell me about the Bitmap pool. -
A memory pool for reusing bitmaps to reduce memory allocation.

Data Saving

Jetpack DataStore Preferences -
A data storage solution that is safer and more efficient than SharedPreferences.

How to persist data in an Android app?
Use SharedPreferences, Room, or DataStore.

What is ORM? How does it work?
Object-Relational Mapping; it abstracts database interactions using objects.

How would you preserve the Activity state during a screen rotation?
Save instance state and use ViewModel.

What are different ways to store data in your Android app?
SharedPreferences, SQLite, Room, files, and DataStore.

Explain Scoped Storage in Android. -
A storage model that restricts access to shared storage to enhance privacy.

How to encrypt data in Android?
Use the Android Keystore System and cryptography libraries.

What is commit() and apply() in SharedPreferences?
commit() is synchronous, apply() is asynchronous.

Look and Feel

What is a Spannable?
An interface for text with markup.

What is a SpannableString?
A string with mutable span information.

What are the best practices for using text in Android?
Use resources for strings, support localization, and prefer Spannables for styling.

How to implement Dark mode in any application?
Use theme resources and the AppCompatDelegate to switch themes.

What are themes and styles? How do you use them?
Themes are collections of attributes; styles define the look of UI components.

How to change the font of the app?
Use custom fonts with the fontFamily attribute or Typeface.

Explain the vector drawable?
Scalable images defined in XML.

Debugging and Testing

How to debug an android application?
Use Android Studio's debugger, logs, and breakpoints.

Tell about the test automation tools available for Android. -
Espresso, UI Automator, and Robolectric.

Additional Questions

What is a ViewModel?
A component for managing UI-related data lifecycle.

What is LiveData?
A lifecycle-aware data holder.

How can you save data in an Android application?
Use Room, SharedPreferences, DataStore, or file storage.

How does Dagger2 work?
It provides dependency injection through compile-time code generation.

Kotlin Coroutines Topics

coroutines:
Coroutines in Kotlin provide a way to write asynchronous code sequentially, making it easier to manage background tasks.

suspend:
The suspend keyword in Kotlin marks a function as suspendable, allowing it to be paused and resumed without blocking the thread.

launch, async-await, withContext:

launch: Starts a new coroutine without returning a result.

async-await: Starts a coroutine that returns a Deferred result, which can be awaited.

withContext: Changes the coroutine context, blocking until the code within it completes.

dispatchers:
Dispatchers in Kotlin Coroutines specify the thread or thread pool on which a coroutine runs, such as Dispatchers.Main for the main thread and Dispatchers.IO for I/O operations.

scope, context, job:

scope: Defines the lifecycle of coroutines, typically bound to a lifecycle component like an activity or ViewModel.

context: Holds information about the coroutine, such as its dispatcher and job.

job: Represents a coroutine's lifecycle, allowing you to cancel it.

lifecycleScope, viewModelScope, GlobalScope:

lifecycleScope: A CoroutineScope tied to the lifecycle of an Activity or Fragment.

viewModelScope: A CoroutineScope tied to a ViewModel, automatically cancelled when the ViewModel is cleared.

GlobalScope: A global CoroutineScope that lives for the entire application's lifetime (use with caution).

suspendCoroutine, suspendCancellableCoroutine:

suspendCoroutine: A low-level API to convert callback-based code to coroutines.

suspendCancellableCoroutine: Similar to suspendCoroutine, but also allows the coroutine to be cancelled.

coroutineScope, supervisorScope:

coroutineScope: Creates a scope and waits for all its children coroutines to complete.

supervisorScope: Similar to coroutineScope, but child coroutines do not cancel their siblings if they fail.

Kotlin Flow API Topics

Flow Builder: Functions like flow {} to create a flow of values.

Operator: Functions that transform or manipulate the data in a flow, like map and filter.

Collector: Functions that collect and handle the emitted values from a flow, like collect {}.

flowOn: Changes the coroutine context of the upstream flow, specifying which dispatcher to use.

dispatchers: Specify the thread or thread pool on which to run the flow, such as Dispatchers.IO for I/O operations.

Operators such as

filter: Emits only values that satisfy a given predicate.

map: Transforms each emitted value using a provided function.

zip: Combines values from multiple flows into pairs.

flatMapConcat: Flattens and concatenates flows emitted by a source flow.

retry: Retries the flow on failure with a specified strategy.

debounce: Emits values after a specified timeout if no new values are emitted during that time.

distinctUntilChanged: Emits values only if they are distinct from the previous one.

flatMapLatest: Flattens and switches to the latest flow emitted by a source flow.

Terminal operators:
Functions that trigger the execution of the flow and collect its values, such as collect, toList, and first.

Cold Flow: Starts emitting values only when collected, each collector gets its own emissions.

Hot Flow: Emits values regardless of collectors, shared among multiple collectors.

StateFlow: A state-holder observable flow that emits the current and new states to collectors.

SharedFlow: A hot flow that emits values to multiple collectors, useful for events.

callbackFlow: Creates a flow from callback-based APIs, suspending on receive channel operations.

channelFlow: A flow builder that uses a channel to emit values, supporting both cold and hot flows.

Other Topics - One Line Answers

Describe SQLite. -
SQLite is a lightweight, relational database engine embedded in Android for local data storage.

Have you used Room-Database?
Yes, Room is an abstraction layer over SQLite, providing a more robust and easier way to manage database interactions.

Can we identify the users who have uninstalled our application?
No, it’s not possible to track uninstalled users directly.

Android Development Best Practices. -
Follow coding standards, optimize performance, secure data, and conduct thorough testing.

React Native vs Flutter. -
React Native uses JavaScript, Flutter uses Dart; both are popular for cross-platform app development.

What are the metrics that you should measure continuously while android application development?
Monitor app launch time, memory usage, CPU usage, network latency, and crash rates.

How to avoid API keys from check-in into VCS?
Use environment variables, properties files, or secret management tools.

How does the Kotlin Multi Platform work?
It allows sharing common code across multiple platforms (iOS, Android, JVM, JS) while maintaining platform-specific code.

How to use Memory Heap Dumps data?
Analyze heap dumps using tools like Android Studio Profiler or MAT to identify memory leaks and optimize memory usage.

How to implement Dark Theme in your app?
Use theme resources and AppCompatDelegate.setDefaultNightMode() to switch between light and dark themes.

How to secure the API keys used in an Android App?
Store API keys in native code using the NDK, encrypted storage, or secure environment variables.

Tell something about memory usage in Android. -
Efficient memory management is crucial to avoid memory leaks and ensure smooth app performance.

Explain Annotation processing. -
It involves generating code or performing compile-time checks based on annotations in the code.

How does the Android Push Notification system work?
It uses Firebase Cloud Messaging (FCM) to deliver notifications from servers to Android devices.

How to show local Notification at an exact time?
Use AlarmManager to schedule the exact time and NotificationManager to show the notification.