Kotlin Flow - Detailed Guide

What is Kotlin Flow?

```
Flow is a cold asynchronous data stream that emits multiple values sequentially.
It is part of Kotlin Coroutines and is useful for reactive programming.
```

Core Concepts

```
- Cold: Doesn't emit until collected.
- Suspendable: emit() and collect() are suspend functions.
- Cancelable: Follows coroutine cancellation.
```

Creating a Flow

```
fun numberFlow(): Flow<Int> = flow {
    for (i in 1..5) {
        emit(i)
        delay(500)
    }
}
```

Flow Operators

```
- map { it * 2 }
- filter { it > 3 }
- collectLatest {}
- flatMapLatest {}
- debounce(), distinctUntilChanged()
```

Cold vs Hot Flows

```
Cold: Emits only on collection. Example: flow {}
Hot: Emits regardless of collectors. Example: StateFlow, SharedFlow
```

StateFlow & SharedFlow

```
StateFlow:
- Always has a value.
- Good for UI state.

SharedFlow:
- Does not hold a value.
- Good for one-time events.
```

Combining Flows

```
flow1.combine(flow2) { a, b -> "$a$b" }
flow1.zip(flow2) { a, b -> ... }
```

Kotlin Flow - Detailed Guide

Error Handling

```
flow.catch { e -> emit(Fallback()) }
    .onCompletion { println("done") }
    .collect { ... }
```

callbackFlow Example

```
fun locationFlow(): Flow<Location> = callbackFlow {
    val listener = LocationListener { location -> trySend(location) }
    locationManager.requestLocationUpdates(..., listener)
    awaitClose { locationManager.removeUpdates(listener) }
}
```

Flow Use Cases

```
- Debounced search
- UI state updates
- Realtime data
- One-shot events
```

Testing Flow

```
runTest {
    val flow = flowOf(1, 2, 3)
    assertEquals(listOf(1, 2, 3), flow.toList())
}
```

Flow Summary Table

```
Feature              | Flow | StateFlow | SharedFlow
---------------------|------|-----------|------------
Cold/Hot             | Cold | Hot       | Hot
Holds Value?         | No   | Yes       | No (configurable)
Use for UI state     | No   | Yes       | No
Multiple collectors  | Yes  | Yes       | Yes
```