Mobile
Innovation
Network

2 min read

Understanding Android Context

Important?

Mobile Innovation Network

Swipe

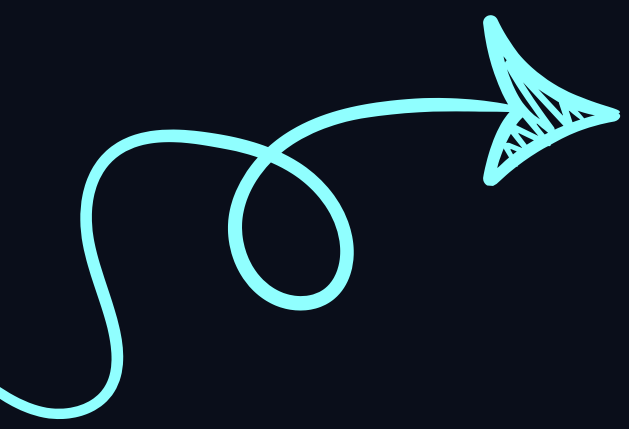
Why it's Important?

- **Accessing Resources like,** Retrieve strings, drawables, and assets.
- **Using System Services like,** Access location, WiFi, and more.
- **Launching Components like,** Start activities, services, or broadcasts.
- **Inflating Layouts like,** Convert XML into UI elements.

Mobile Innovation Network

Swipe

Different Types of Context in Android



1. Application Context

- Tied to the entire application lifecycle.
- Ideal for accessing system services and background operations.
- Should not be used for UI-related tasks.

```
class MyApp : Application() {
    override fun onCreate() {
        super.onCreate()
        val appContext = applicationContext
        // Use appContext for global operations like initializing libraries
    }
}
```

2. Activity Context

- **Activity-Specific:** Follows activity lifecycle.
- **UI Tasks:** Handles dialogs, toasts, and layout inflation.
- **Memory Caution:** Avoid passing to long-lived objects.

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val activityContext = this
        val inflater = LayoutInflater.from(activityContext)
        val view = inflater.inflate(R.layout.activity_main, null)
        setContentView(view)
    }
}
```

Common Mistakes

- Storing Activity Context in static fields → **Memory leaks.**
- Using Application Context for UI → **Crashes.**
- Use the right Context for each task.

Cause Memory Leaks

- Static Activity Context → **Retains entire Activity.**
- Inner classes hold outer **class references**.
- Background tasks with Context → **High memory use.**
- Use Application Context & **clear references**.

Mobile Innovation Network

Swipe

Best Practices for
Using Context

- **Use Application** Context for global operations and background tasks.
- **Use Activity** Context only for UI-related operations.
- **Avoid passing Context** to objects that persist beyond their necessary lifecycle.
- **Use Weak Reference** when storing Context in long-lived objects.

Mobile Innovation Network

Swipe

Context in Jetpack Compose

✨ Still required for some operations.

🔷 Use LocalContext.current in Compose.

✅ Prefer ViewModels for state management.

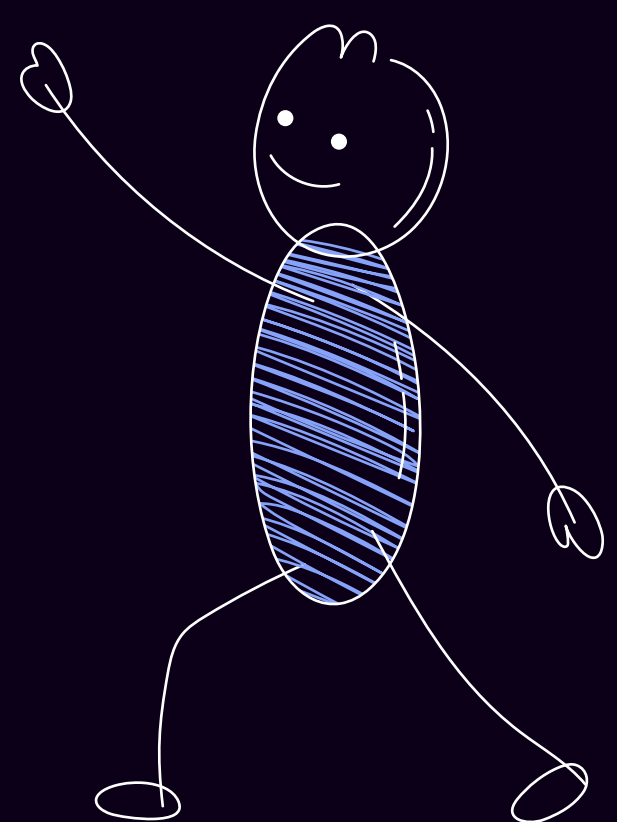
```
@Composable
fun MyComposable() {
    val context = LocalContext.current
    Button(onClick = {
        Toast.makeText(context, "Hello from Compose!", Toast.LENGTH_SHORT).show()
    }) {
        Text("Show Toast")
    }
}
```

Key Takeaways: 🔑

👉 Context is essential for Android development.

👉 Misuse can cause crashes & memory leaks.

👉 Follow best practices for safe usage.



Thank you for your Attention!