Step it up: Compose for Desktop
Jetpack Compose
Jetpack Compose

- Modern Toolkit
- API inspired by React and Flutter
- Declarative and Reactive UI Framework
- Simplify UI development
- Built using Kotlin Language
- Currently in ALPHA 😲
Jetpack Compose
@Composable
fun helloWorld() {
    Text(text = "Hello World!")
}
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            helloWorld()
        }
    }
}
@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    MyApplicationTheme {
        helloWorld()
    }
}
Jetpack Compose

Compose Compiler Plugin + Jetpack Compose
Compose for Desktop
Compose for Desktop

- Everything Jetpack Compose for Android has
- Simplified Desktop UI Development
- Targets only JVM right now
- First public milestone was in Nov 2020
- Currently in **ALPHA** 😲
Compose for Desktop

- Everything Jetpack Compose for Android has
- Simplified Desktop UI Development
- Targets only JVM right now
- First public milestone was in Nov 2020
- Currently in **ALPHA**
- ..is also called “Jetpack Compose for Desktop”
Compose for Desktop

New Project

Name: ComposeDesktopExample
Location: /home/zion/IdeaProjects/ComposeDesktopExample
Project Template: Kotlin Application
Build System: Gradle
Project JDK: adopt-openjdk-11 version 11.0.9

A Compose application targeting the desktop JVM platform (Windows, Linux, macOS)
import androidx.compose.material.Text
import androidx.compose.runtime.Composable

@Composable
fun helloWorld() {
    Text(text = "Hello World!")
}
import androidx.compose.material.Text
import androidx.compose.runtime.Composable

@Composable
fun helloWorld() {
    Text(text = "Hello World!"
)}
import androidx.compose.desktop.Window

fun main() = Window {
    helloWorld()
}
import androidx.compose.desktop.Window

fun main() = Window {
    helloWorld()
}

Note: There is no setup right now to Preview, like in Android! 😞
`compose.desktop {
    application {
        mainClass = "MainKt"
        nativeDistributions {
            targetFormats(TargetFormat.Dmg, TargetFormat.Msi, TargetFormat.Deb)
            packageName = "ComposeDesktopExample"
        }
    }
} `
compose.desktop {
  application {
    mainClass = "MainKt"
    nativeDistributions {
      targetFormats(TargetFormat.Dmg, TargetFormat.Msi, TargetFormat.Deb)
      packageName = "ComposeDesktopExample"
    }
  }
}
enum class TargetFormat(
    internal val id: String, private vararg val compatibleOSs: OS) {
    AppImage("app-image", *OS.values()),
    Deb("deb", OS.Linux),
    Rpm("rpm", OS.Linux),
    Dmg("dmg", OS.MacOS),
    Pkg("pkg", OS.MacOS),
    Exe("exe", OS.Windows),
    Msi("msi", OS.Windows);
    ...}
# Binaries are generated at
# Path: `{projectName}/build/compose/binaries/main/{packageType}`

`./gradlew package`
# Binaries are generated at
# Path: {projectName}/build/compose/binaries/main/{packageType}
Why?

- Traditional Desktop UI frameworks
  - SWING
  - AWT
  - Callback driven
- Electron
- Qt, GTK, etc.
So what's the magic?
So what's the magic?

- **Skia**
  - 2D rendering library
  - Open sourced under BSD
  - Maintained by Google
  - Compose is using Skija ([Java bindings for Skia](#))
What’s next?
What’s Next?

- First Jetpack Compose for Android, then Compose for Desktop
- Testing setup
  - Possible: Screenshot Testing, Mocking
- Should work in theory on GraalVM
- Target native platforms
What’s Next?

https://github.com/JetBrains/compose-jb/tree/master/examples
Links

https://skia.org
https://github.com/JetBrains/skija
https://github.com/jetbrains/compose-jb
Credits

○ Presentation template by SlidesCarnival
Thanks!

Any questions?

Nishant Srivastava
www.nisrulz.com