Kotlin Compose for Web

18.10.2023 | Fabian Grutsch

Agenda

- What is Kotlin Compose Multiplatform?
- How does Jetpack Compose work?
- Setup & Demo Application

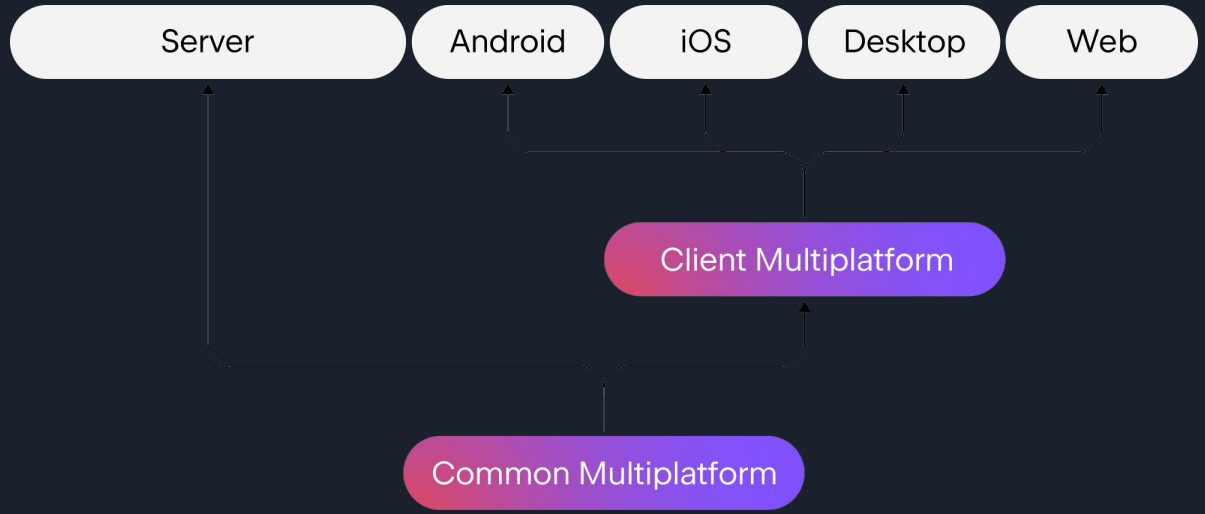
What is Kotlin Compose Multiplatform?

- based on Jetpack Compose (Android's modern toolkit for building UI)

- declarative framework for sharing UIs across multiple platforms with Kotlin

- developed by JetBrains and open-source contributors

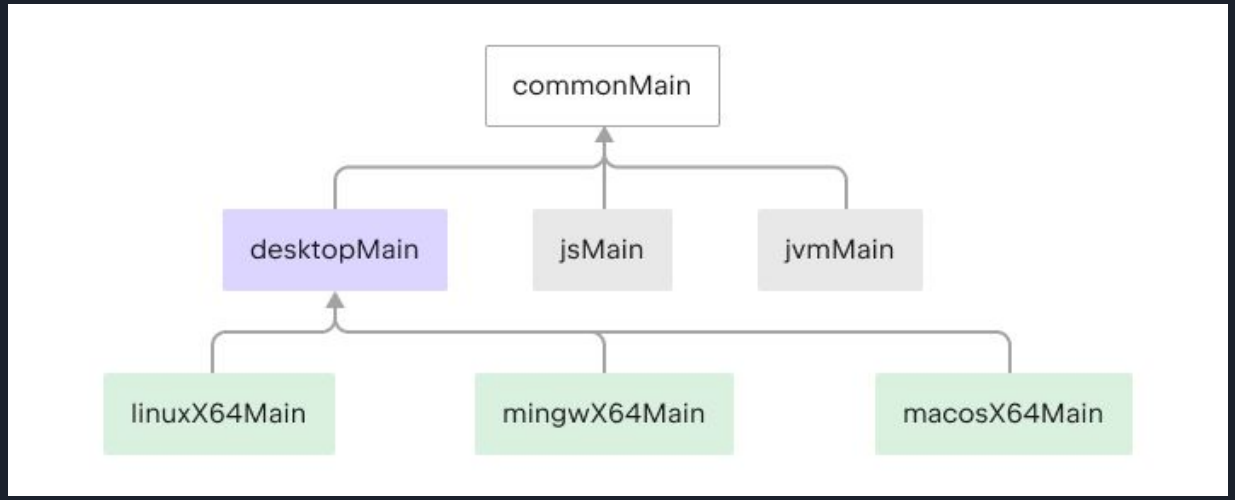
Kotlin Compose Multiplatform - Platforms

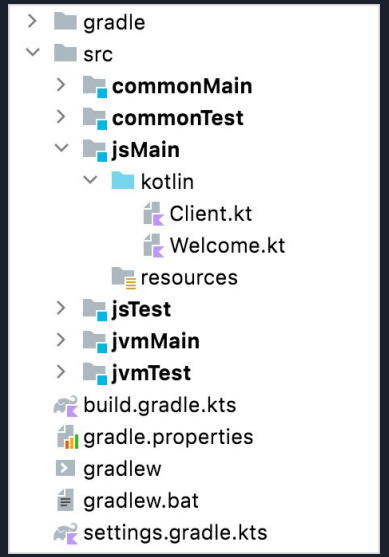


Notes:

- “Web” is based on Kotlin/Wasm
- for HTML/CSS use Compose HTML (targets Kotlin/JS)

Kotlin Compose Multiplatform - Source Sets





Jetpack Compose

- built around composable functions
- define UI programmatically
- add the ***@Composable*** annotation to the function name
- hierarchical UI - call composable functions from other composable functions

```
@Composable
fun Greeting(name: String) {
    Div { Text("Hello $name!") }
}
```

Recomposition

- is the process of calling your composable functions again when inputs change
- Compose framework can intelligently recompose only the components that changed
- keep free of side-effects (e.g. writing to shared property, etc.)

```
@Composable
fun ClickCounter(clicks: Int, onClick: () -> Unit) {
    Button(
        attrs = { onClick { onClick() } }
    ) {
        Text("Clicked $clicks times")
    }
}
```

Managing state

- ***mutableStateOf*** creates an observable ***MutableState<T>*** (observable type integrated with the compose runtime)
- ***remember*** API stores object in memory and value is returned during recomposition
- more types of states (e.g. ***collectAsState***, *etc.)*

```
@Composable
fun TextInput() {
    var input by remember { mutableStateOf( value: "") }

    if (input.isNotEmpty()) {
        Text("Typed: $input")
    }

    Input(InputType.Text) {
        value(input)
        onInput { input = it.value }
    }
}
```

Effects

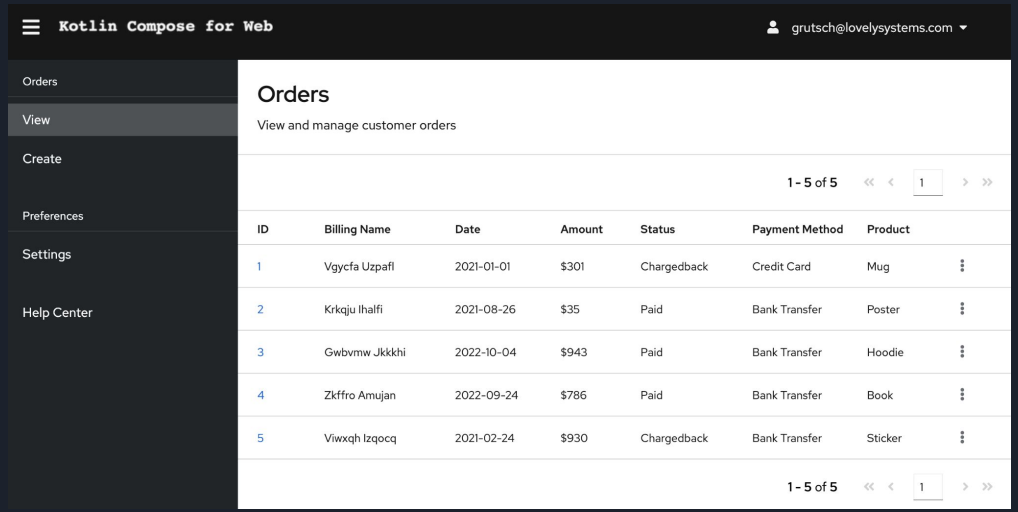
- ***LaunchedEffect*** run suspend functions safely in the scope of a composable
- ***derivedStateOf*** converts one or multiple state objects into another state
- more side-effect functions

```
@Composable
fun Example(key: String) {
    LaunchedEffect(key) {
        delay( timeMillis: 1000) // Delay is suspending
        println("LaunchedEffectExample")
    }
}
```

```
@Composable
fun Example() {
    var a by mutableStateOf( value: 0)
    var b by mutableStateOf( value: 0)
    val sum by derivedStateOf { a + b }
    // Do something with sum (e.g. display it)
}
```

Setup & Demo Application

- uses the Compose HTML library
- common / platform independent code in its own shared package
- ViewModels to expose state to the UI and encapsulate business logic
- designed with PatternFly (Open Source design system by Red Hat)



References

- Slides & Code: https://github.com/fgrutsch/kotlin-compose-for-web
- Compose Multiplatform: https://www.jetbrains.com/lp/compose-multiplatform/
- Compose HTML: https://github.com/JetBrains/compose-multiplatform/#compose-html
- Jetpack Compose: https://developer.android.com/jetpack/compose