Modern development with Jetpack Compose

Fabio Giovanazzi (@Stypox) – Speck&Tech retreat 2023

Fabio
Giovanazzi

@Stypox
github.com/Stypox

University of Trento

NewPipe

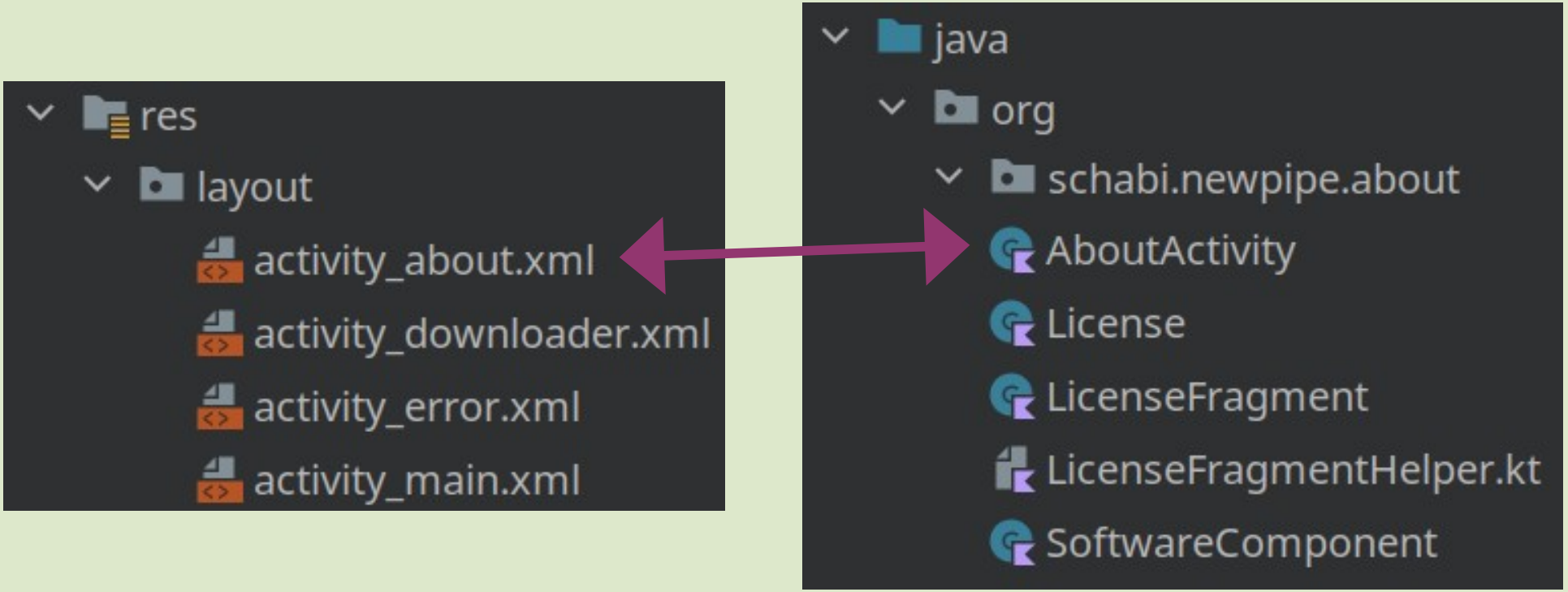
Dicio assistant

Tridenta

MindsHub APS

Traditional Android development

Traditional Android development



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <TextView android:id="@+id/my_text_view" />

    <Button android:id="@+id/my_button" />

</LinearLayout>
```

```
setContentView(R.layout.activity_about)

val myTextView = findViewById<TextView>(R.id.my_text_view)
val myButton = findViewById<Button>(R.id.my_button)

myTextView.text = "My text view text"
myButton.setOnClickListener { it: View!
    print("Text clicked!")
}
```

Declarative
approach

Imperative vs **Declarative**

(1) Create initial view **tree**

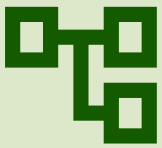
(2) Keep tree in **sync** with state

```
UI =
f(state)
```

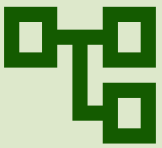
State

single source of truth

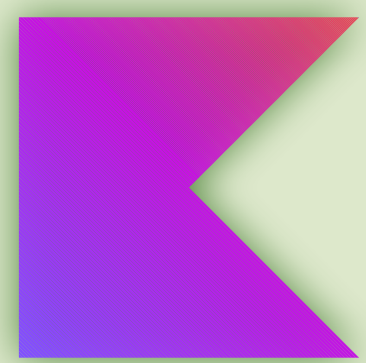
UI is **redrawn/recomposed** when it changes*



*frameworks perform **diffs** to make this fast



Jetpack
Compose

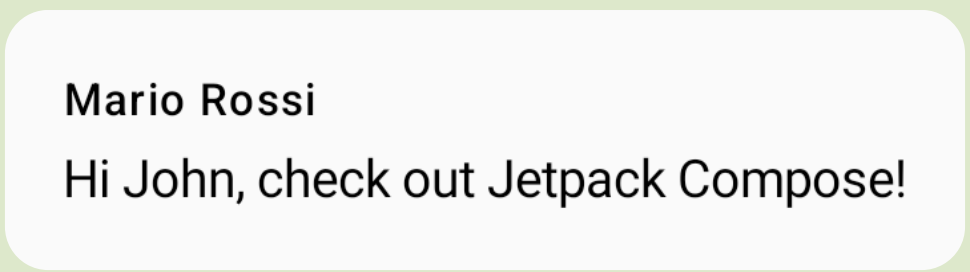


```
@Composable
fun ChatMessage(data: MessageData) {
    Column(modifier = ... ) {
        Text(
            data.messageAuthor,
            style = ...,
        )

        Spacer(modifier = Modifier.height(4.dp))

        Text(data.messageText)
    }
}
```

```
ChatMessage(
    data = MessageData(
        messageText = "Hi John, ...",
        messageAuthor = "Mario Rossi",
    )
)
```



Traditional vs Compose


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout>

    <TextView android:id="@+id/my_text_view" />

    <Button android:id="@+id/my_button" />

</LinearLayout>
```

```
setContentView(R.layout.activity_about)

val myTextView = findViewById<TextView>(R.id.my_text_view)
val myButton = findViewById<Button>(R.id.my_button)

myTextView.text = "My text view text"
myButton.setOnClickListener { it: View!
    print("Text clicked!")
}
```

```
@Composable
fun ChatMessage(data: MessageData) {
    Column(modifier = ... ) {
        Text(
            data.messageAuthor,
            style = ...,
        )

        Spacer(modifier = Modifier.heig

        Text(data.messageText)
    }
}
```

- 👍 Fewer **state** bugs
- 👍 **Reusability**
- 👍 Less **boilerplate**

And more...

Hands on!

- download **Android Studio**: https://developer.android.com/studio
- open Android Studio and download the **demo app** we'll work on by pressing on "`File -> New -> Project from version control...`" and inserting this URL: https://github.com/Stypox/compose-demo
- wait until the project loads, then **build** the project by pressing on "`Build -> Make Project`"
- *(optional, UI previews can be viewed even without)* download an **emulator** using this tutorial: https://developer.android.com/studio/run/managing-avds#createavd

- Full Jetpack Compose **course**
  https://developer.android.com/courses/jetpack-compose/course
- **View Models** to separate state and business logic from the UI
  https://developer.android.com/codelabs/basic-android-kotlin-compose-viewmodel-and-state
- **Dependency injection** for flexible and modular code
  https://developer.android.com/training/dependency-injection
- **Material Design 3**
  https://developer.android.com/jetpack/compose/designsystems/material3
- **Navigation** between screens
  https://developer.android.com/jetpack/compose/navigation
- Compose **multiplatform**
  https://www.jetbrains.com/lp/compose-multiplatform/

Used in Tridenta