

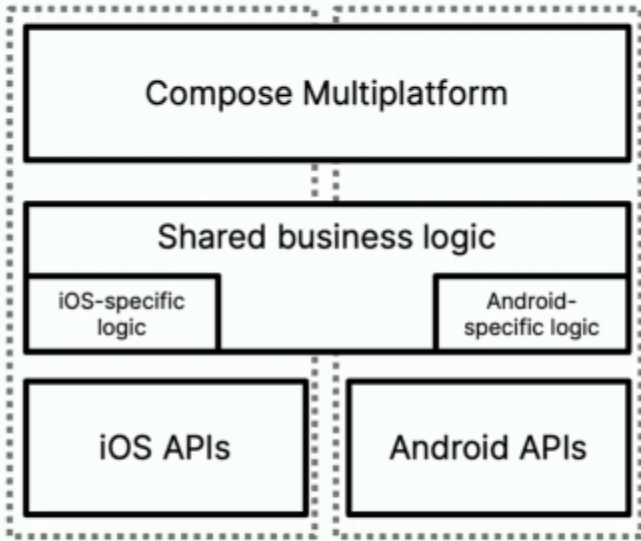



ZeroPage 정모 발표

Jetpack Compose 한번 써본 후기









```
graph TD; subgraph " "; C[Compose Multiplatform] --- S[Shared business logic]; S --- iOSL[iOS-specific logic]; S --- AndroidL[Android-specific logic]; iOSL --- iOSA[iOS APIs]; AndroidL --- AndroidA[Android APIs]; end
```

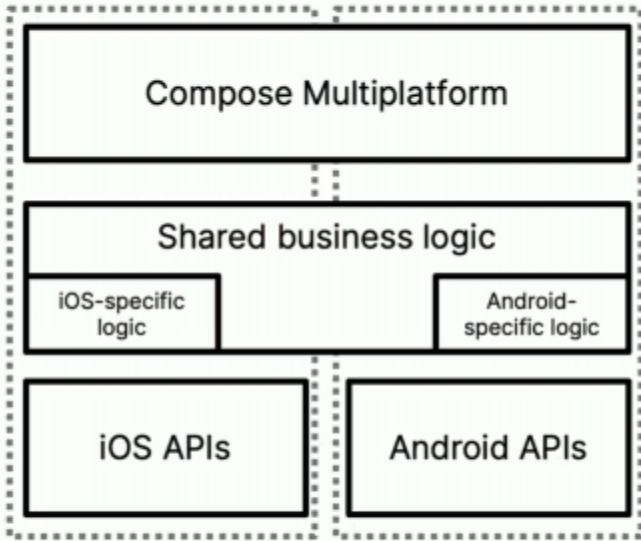


THE KOTLINCONF'23 LIVESTREAM
IS BROUGHT TO YOU BY [GETSTREAM.IO](https://getstream.io)




KOTLINCONF'23
AMSTERDAM







```
graph TD; subgraph " "; C[Compose Multiplatform] --- S[Shared business logic]; S --- iOSL[iOS-specific logic]; S --- AndroidL[Android-specific logic]; iOSL --- iOSA[iOS APIs]; AndroidL --- AndroidA[Android APIs]; end
```



THE KOTLINCONF'23 LIVESTREAM
IS BROUGHT TO YOU BY [GETSTREAM.IO](https://getstream.io)



KOTLINCONF'23
AMSTERDAM

WASM 

23-24 MARCH | BARCELONA

WASM I/O 2023



WASM 

zal.im/wasmio

#WASMIO23

Compose Multiplatform Kotlin/Wasm Demo

[CircleOfCirclesWithSettings](#)

[CircleOfCircles](#)

[SeedOfLife](#)

[Spiral](#)

[DrawingRandomShapes](#)

[DrawingStrokes](#)

Settings

Draw inner orbit



Draw outer orbit



Orbits radius: 447.30



Steps for inner orbit: 3



Steps for outer orbit: 5



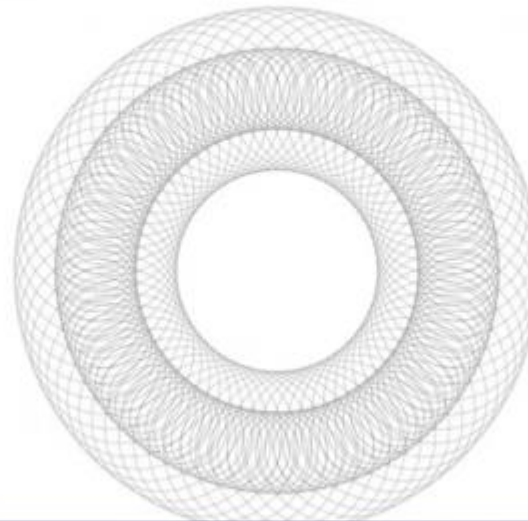
Inner orbit circles radius: 100.0



Outer orbit circles radius: 200.0

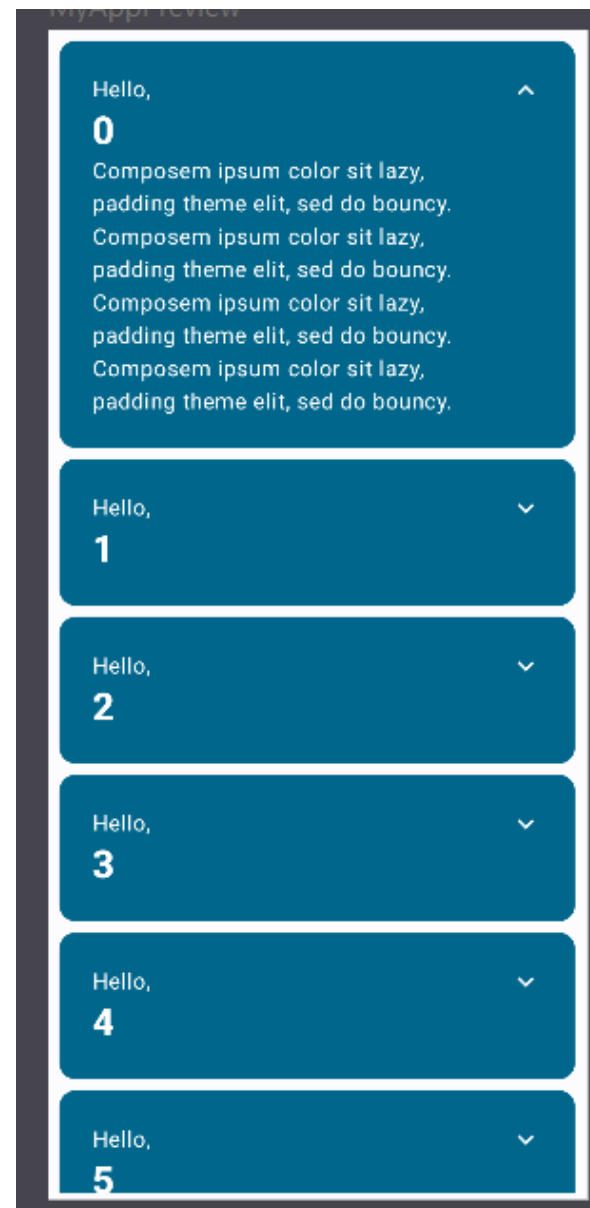
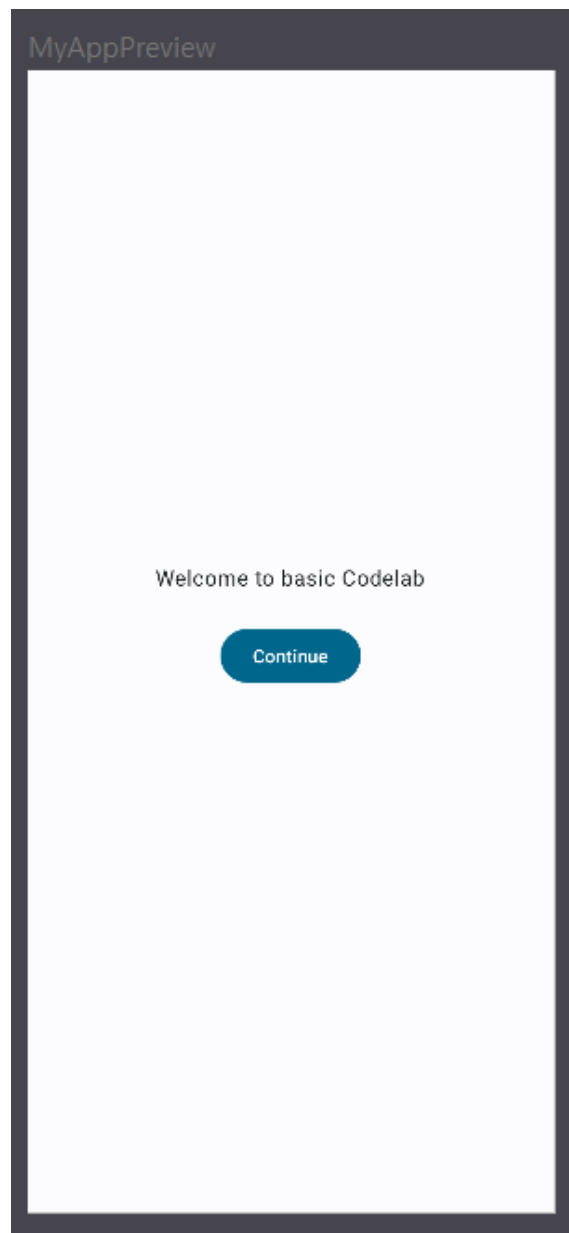
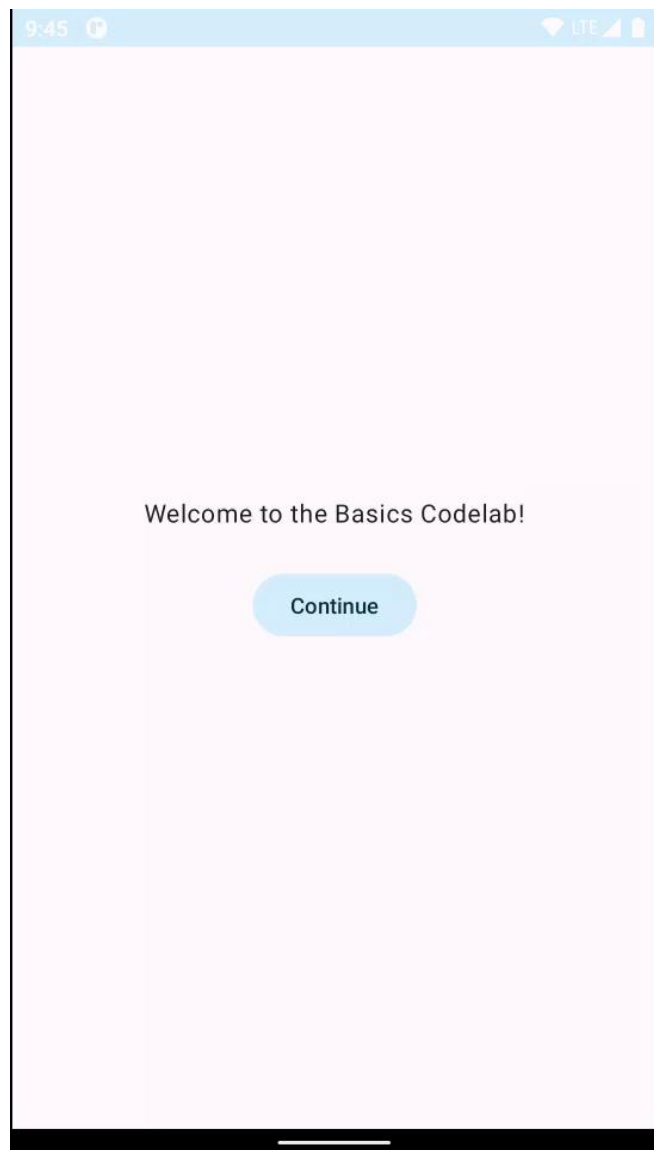


Random: 0.10



- Flutter라는 좋은 멀티플랫폼 프레임워크가 있었기 때문에 Jetpack Compose는 서서히 사장되어갈줄 알았음.
- 갑자기 4월달에 Jetpack Compose 멀티플랫폼 지원 발표. 거기다가 웹까지 베타로 시연했음.
- Flutter나 파려했는데, 갑자기 끌려서 한번 시작해봤습니다.

- Jetpack Compose란?
- 안드로이드의 UI를 쉽게 디자인 하고 빌드하기 위한 라이브러리
- 레이아웃의 쉬운 작성과 고성능을 위해서 디자인됨.



- 원래의 xml을 사용한 디자인에서는 동적으로 TextView에 text를 출력할때 다음의 과정을 거쳤다.
- Layout파일을 작성.
- Layout 파일을 inflat하고, Layout안의 Textview를 찾음.
- TextView에 text출력.

activity_main.xml MainActivity.kt

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:id="@+id/text_main"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent"
17         tools:text="Hello World!" />
18
19  </androidx.constraintlayout.widget.ConstraintLayout>

```

activity_main.xml MainActivity.kt

```

1  package com.example.myapplication
2
3  import ...
4
5
6
7  class MainActivity : AppCompatActivity() {
8      override fun onCreate(savedInstanceState: Bundle?) {
9          super.onCreate(savedInstanceState)
10         setContentView(R.layout.activity_main)
11
12         val textView = findViewById<TextView>(R.id.text_main)
13         textView.text = "Kotlin World Blog"
14     }
15 }

```

```

buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}

buildFeatures {
    viewBinding true
}

compileOptions {
    sourceCompatibility JavaVersion.VERSION_1_8
    targetCompatibility JavaVersion.VERSION_1_8
}

kotlinOptions {
    jvmTarget = '1.8'
}

```

```

package com.example.myapplication

import ...

class MainActivity : AppCompatActivity() {

    private lateinit var binding : ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        binding = ActivityMainBinding.inflate(layoutInflater)

        binding.clickbutton.setOnClickListener { it: View!
            binding.textView.text = "1"
        }

        setContentView(binding.root)
    }
}

```

```
package com.example.jetpackexample

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView {
            Text(text: "Kotlin World Blog")
        }
    }
}
```

- 하지만 Jetpack Compose에서는 리소스의 출력과 Textview를 그려주는 부분이 합쳐져있음.
- 딱 한줄로 텍스트를 출력가능해진다.

- Xml과는 달리, Constraint Layout을 쓸 필요가 없어졌다.
- Xml에서 Constraint Layout을 쓰는 이유는 성능의 향상을 위해서 인데, Tree구조가 아닌 Flat구조인 compose는 아무리 중첩을 해도 중첩을 하는게 아닌 걸로 처리가 되기 때문에.

- 또한 복잡하게 xml을 왔다리갔다리 할 필요가 없어졌다. 바로 MainActivity단에서 처리가 가능해졌기 때문에.
- 특히 맨날 디자인 한다고 복잡한 xml파일들을 들여다 볼 필요가 확연히 줄어들음. Compose에서 xml을 다루는 영역은 거의 없다싶이 함.

```

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Sutdy3Theme {
                Greetings(modifier = Modifier.fillMaxSize())
            }
        }
    }
}

```

```

@Composable
fun OnboardingScreen(
    onContinueClicked: () -> Unit,
    modifier: Modifier = Modifier) {
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        this: ColumnScope
        Text(text = "Welcome to basic CodeLab")
        Button(
            modifier = Modifier.padding(vertical = 24.dp),
            onClick = onContinueClicked
        ) {
            this: RowScope
            Text(text = "Continue")
        }
    }
}

```

- 또 RecyclerView가 LazyColumn으로 대체되었는데 이게 매우 편리해졌다.
- 기존 RecyclerView는 Adapter, LayoutManager등등 여러 정보가 필요했었는데, LazyColumn은 그런거 필요없이 데이터만 넣으면 되서 매우 편리해졌다.


```
@Composable
private fun Greetings(
    modifier: Modifier = Modifier,
    names: List<String> = List(size: 1000) {"$it"}
) {
    LazyColumn(modifier = modifier.padding(vertical = 4.dp)) { this: LazyListScope
        items(items = names) { name ->
            Greeting(name = name)
        }
    }
}
```

- 써보고 느낀점
- 레이아웃 파일을 관리안해도 되는게 매우 좋았고, RecyclerView등 등 원래 불편했던 점들을 개선시킨게 괜찮았다.
- 하지만 레이아웃을 바로 보기가 뻑세다는 점과, 레이아웃을 마우스로 배치하는 것이 불가능하다는게 단점..
- 또한 자료가 좀 부족하다. 안드로이드 공식 Documentation이 잘되어있기 하지만..

- 또한 처음에 이야기했던 멀티플랫폼도 아직은 지원이 완벽하기 않기 때문에 지금은 Only Android라고 생각하는 것이 좋을듯..
- 특히 코드를 수정안해도 작동할 수는 있지만, Android만의 전용 라이브러리가 호환이 안되기 때문에, 몇몇기능들은 클래스를 새로 만들어서 구현해야 한다는 문제점이 있었음.

- Compose vs Xml
- Xml문법이 익숙한 사람이 아닌 이상, 프로그래머들이 접하기 편한 것은 Compose라고 생각함.
- 특히 구글이 Compose를 밀어주고 있고, 자사의 앱들도 Compose로 대체중이기 때문에, 처음 안드로이드 개발을 접하는 사람들은 Compose가 확실한 우위를 점할거 같다는 생각..

- Compose vs Flutter
- 멀티플랫폼으로 갈거면 Flutter가 지금은 맞는거같다.. 아직 안드로이드 빼곤 다른 플랫폼들은 나온지도 얼마 안되었을 뿐더러 불편한점이 조금은 있기 때문에..
- 근데 Only Android로 개발할거면, Compose이 기반이 Kotlin이라는 구글에서 밀어주는 언어라는 것이 큰 매리트로 작용할수도..