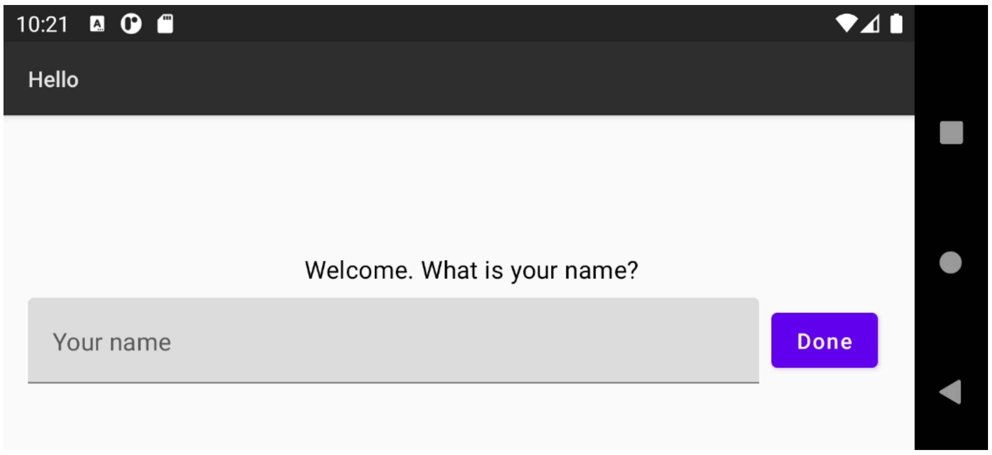
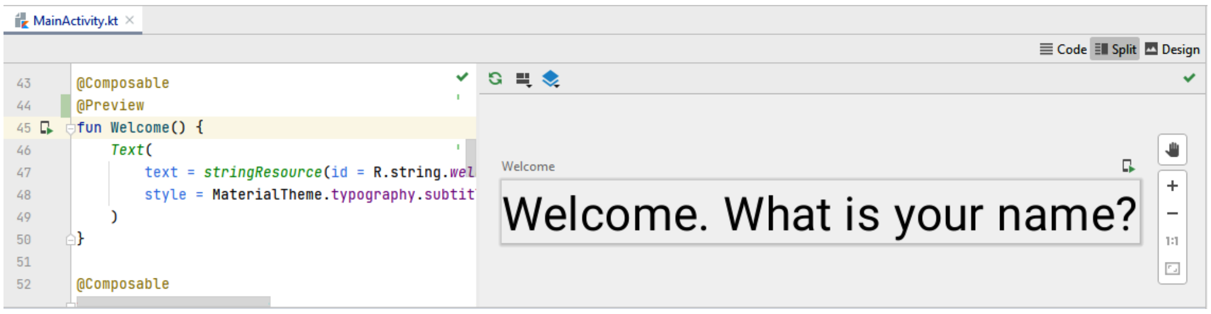
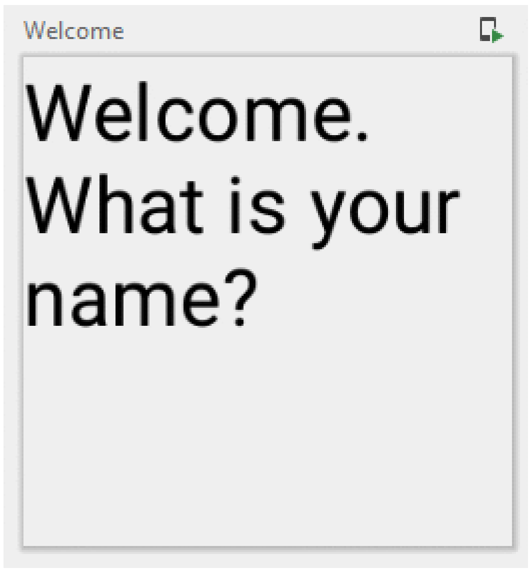
Chapter 01: Building Your First Compose App



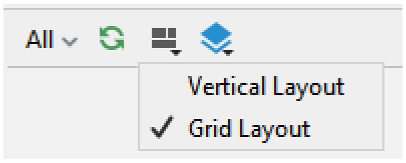
Welcome. What is your name?

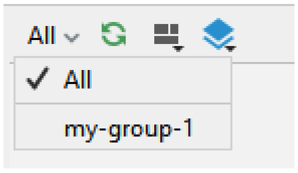


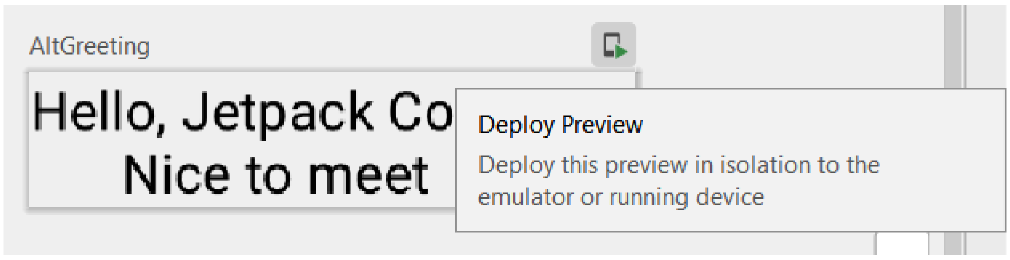
```
@Composable
@Preview
fun Welcome() {
    Text(
        text = stringResource(id = R.string.wel
        style = MaterialTheme.typography.subtit
    )
}

@Composable
```
Welcome. What is your name?



Welcome. What is your name?

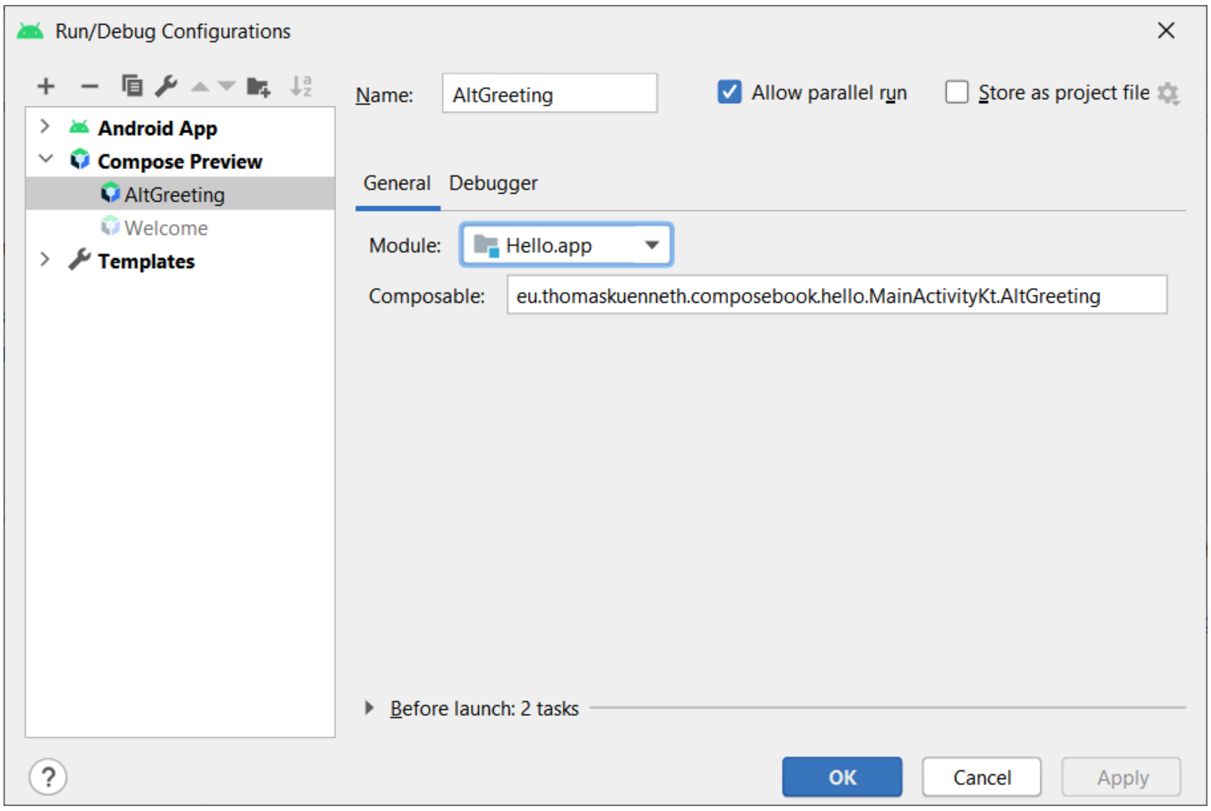






Deploy Preview

Deploy this preview in isolation to the emulator or running device


Run/Debug Configurations

- Android App
- Compose Preview
  - AltGreeting
  - Welcome
- Templates

Name: AltGreeting

[✓] Allow parallel run

[ ] Store as project file

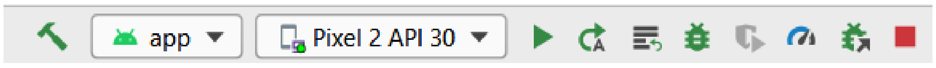
General | Debugger

Module: Hello.app

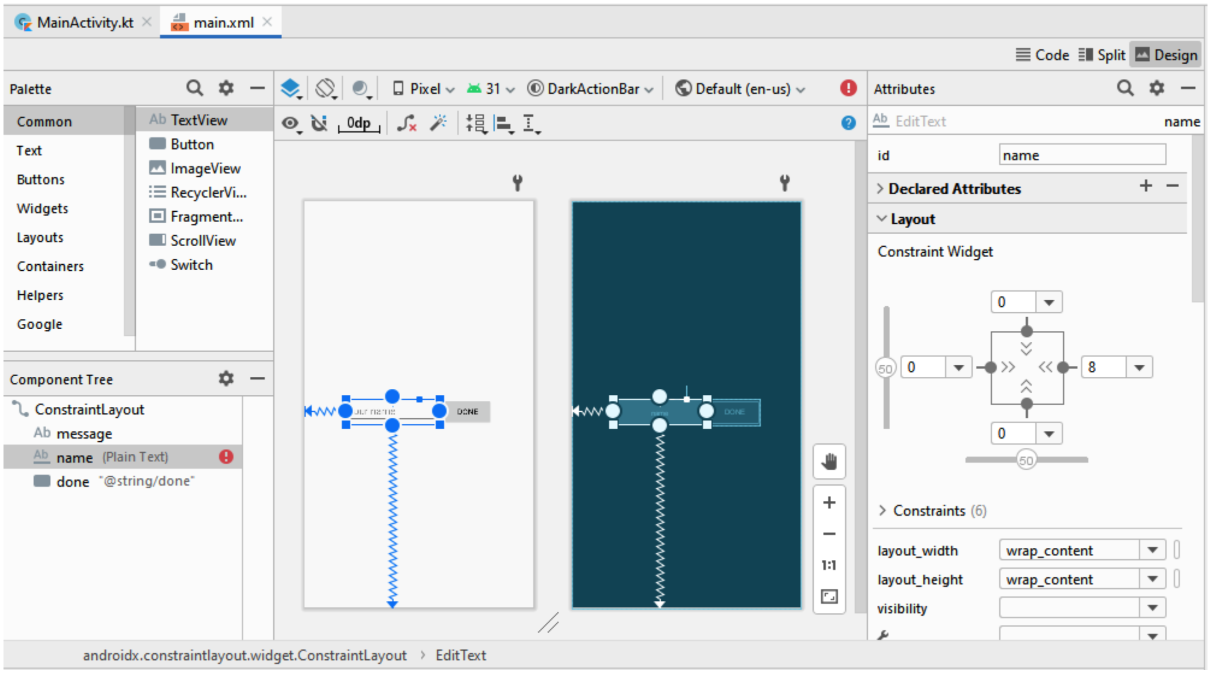
Composable: eu.thomaskuenneth.composebook.hello.MainActivityKt.AltGreeting

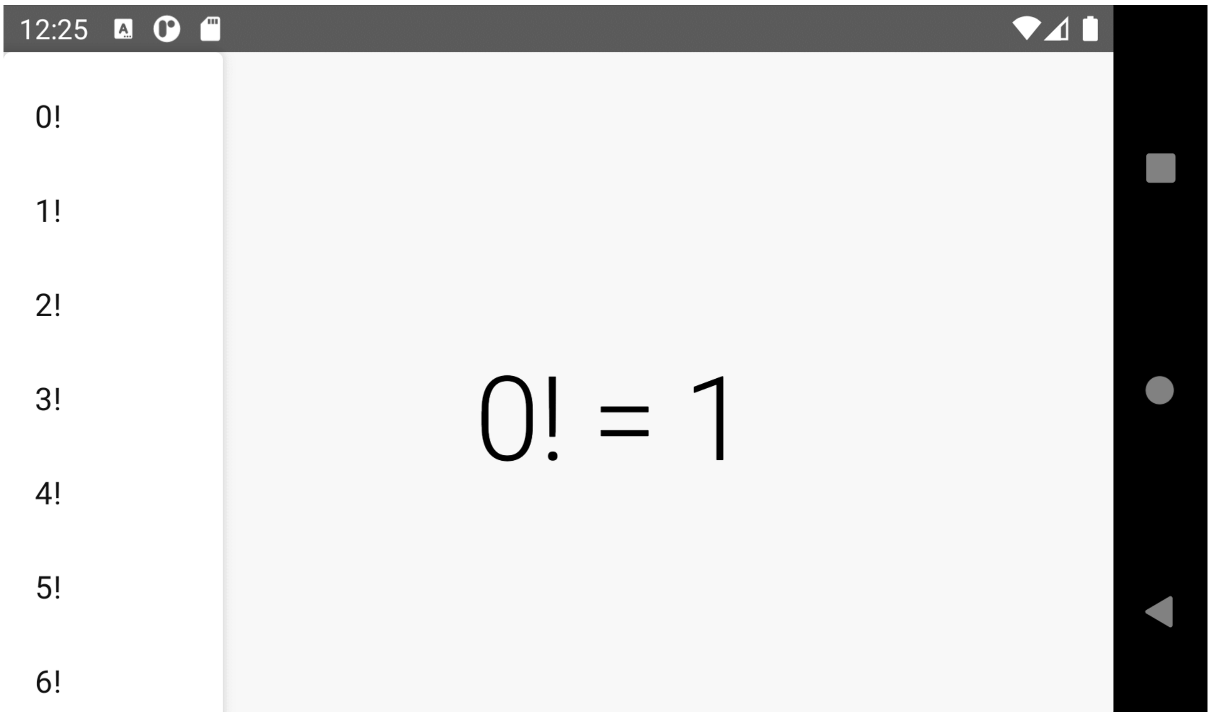
Before launch: 2 tasks

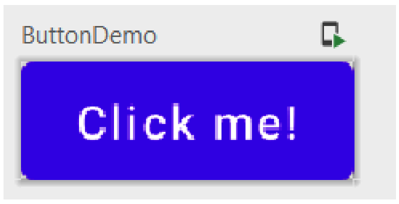
OK | Cancel | Apply

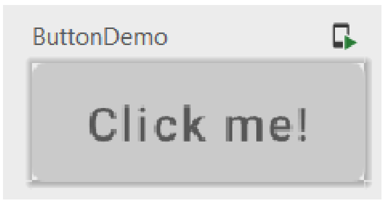


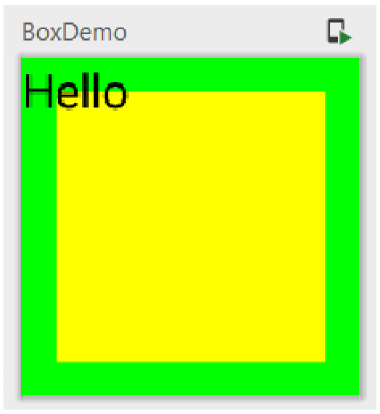
Chapter 02: Understanding the Declarative Paradigm



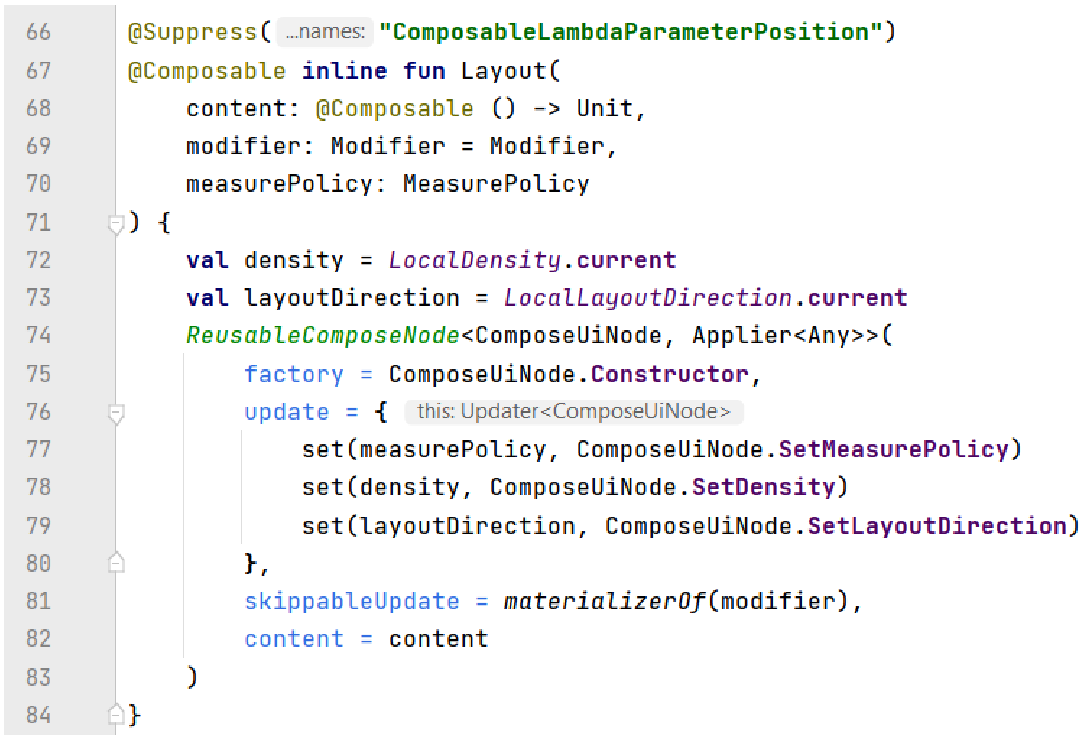



ButtonDemo
Click me!

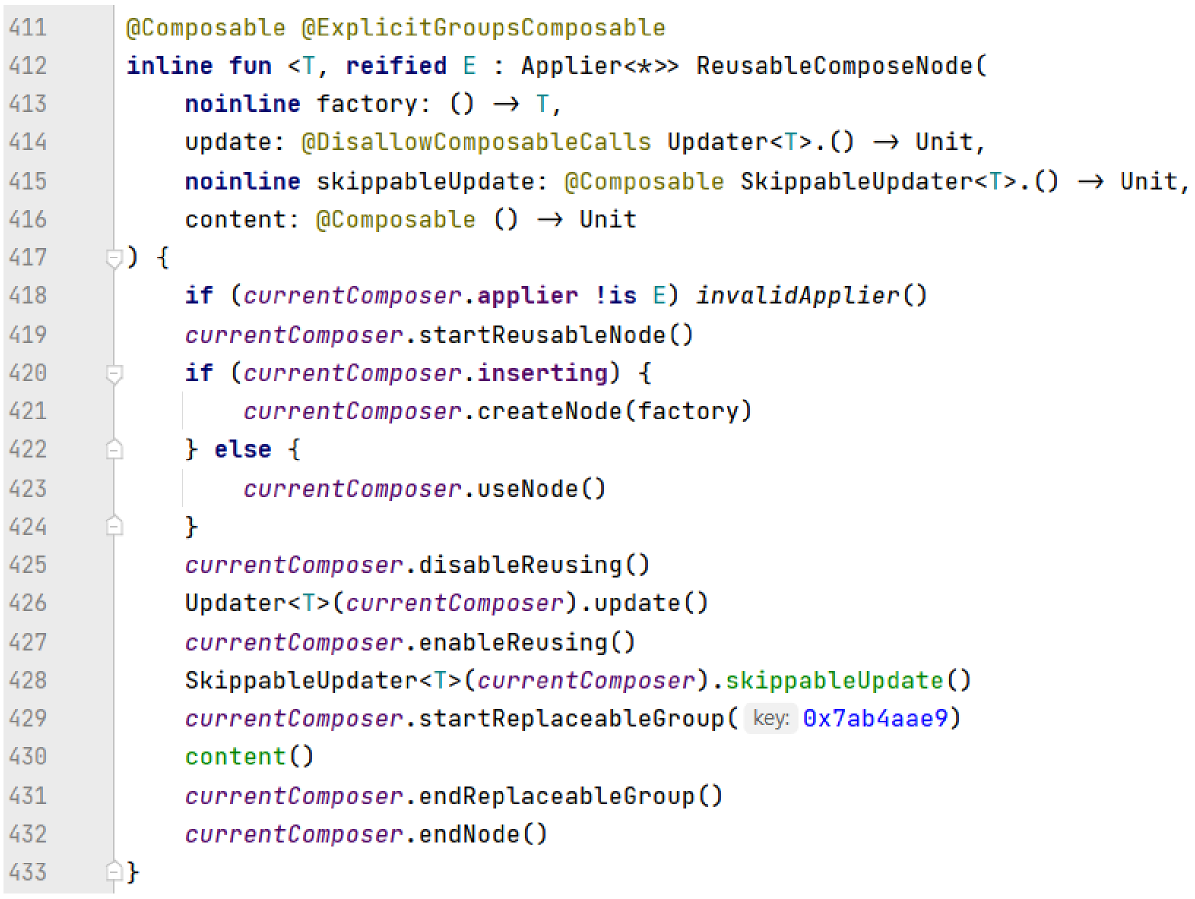

ButtonDemo
Click me!


BoxDemo
Hello

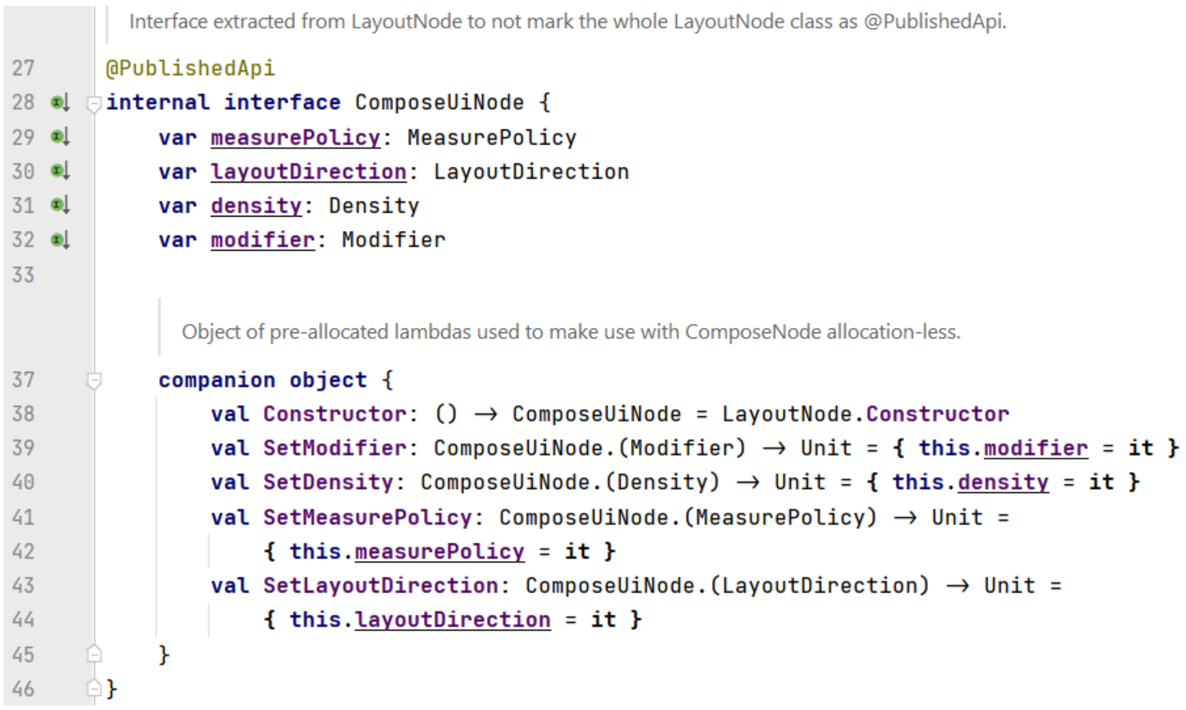
Chapter 03: Building Your First Compose App



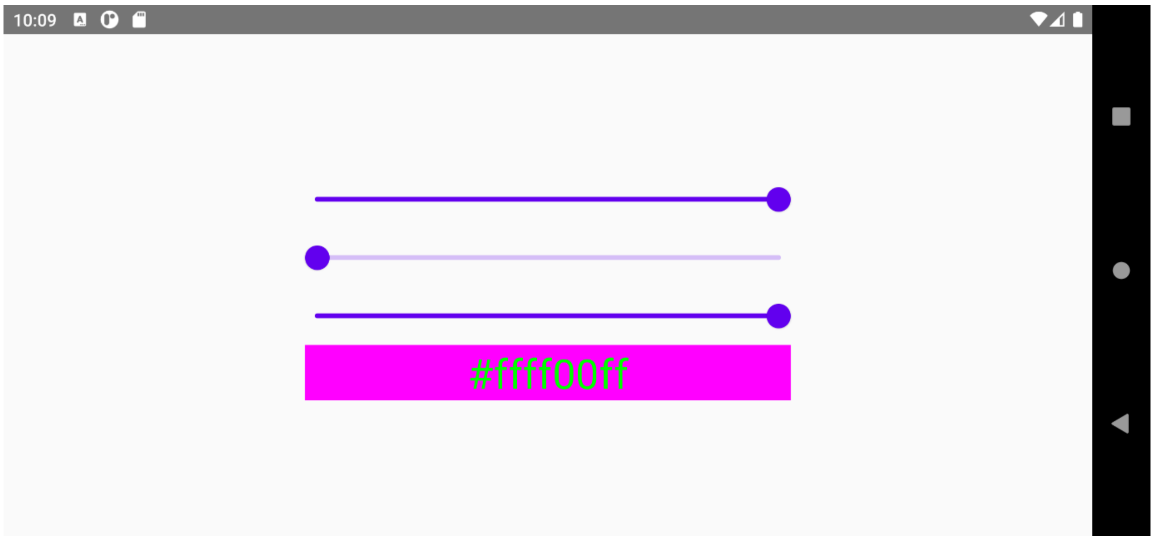
```kotlin
@Suppress( ...names: "ComposableLambdaParameterPosition")
@Composable inline fun Layout(
    content: @Composable () -> Unit,
    modifier: Modifier = Modifier,
    measurePolicy: MeasurePolicy
) {
    val density = LocalDensity.current
    val layoutDirection = LocalLayoutDirection.current
    ReusableComposeNode<ComposeUiNode, Applier<Any>>(
        factory = ComposeUiNode.Constructor,
        update = {  this: Updater<ComposeUiNode>
            set(measurePolicy, ComposeUiNode.SetMeasurePolicy)
            set(density, ComposeUiNode.SetDensity)
            set(layoutDirection, ComposeUiNode.SetLayoutDirection)
        },
        skippableUpdate = materializerOf(modifier),
        content = content
    )
}
```

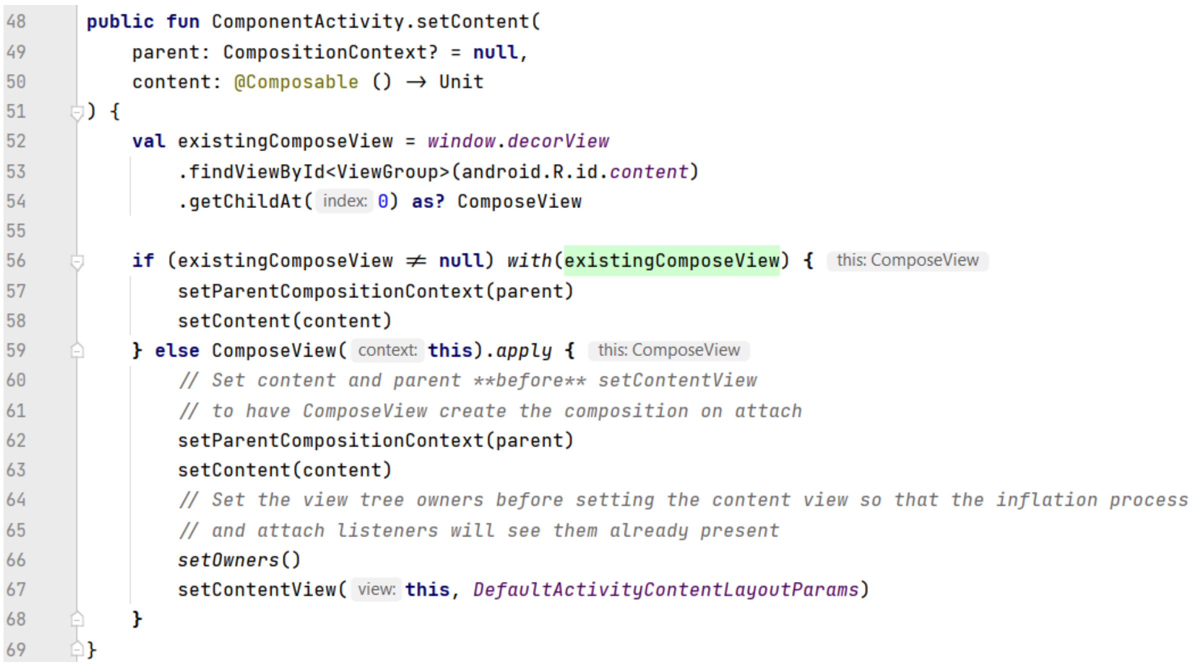


```kotlin
@Composable @ExplicitGroupsComposable
inline fun <T, reified E : Applier<*>> ReusableComposeNode(
    noinline factory: () → T,
    update: @DisallowComposableCalls Updater<T>.() → Unit,
    noinline skippableUpdate: @Composable SkippableUpdater<T>.() → Unit,
    content: @Composable () → Unit
) {
    if (currentComposer.applier !is E) invalidApplier()
    currentComposer.startReusableNode()
    if (currentComposer.inserting) {
        currentComposer.createNode(factory)
    } else {
        currentComposer.useNode()
    }
    currentComposer.disableReusing()
    Updater<T>(currentComposer).update()
    currentComposer.enableReusing()
    SkippableUpdater<T>(currentComposer).skippableUpdate()
    currentComposer.startReplaceableGroup( key: 0x7ab4aae9)
    content()
    currentComposer.endReplaceableGroup()
    currentComposer.endNode()
}
```

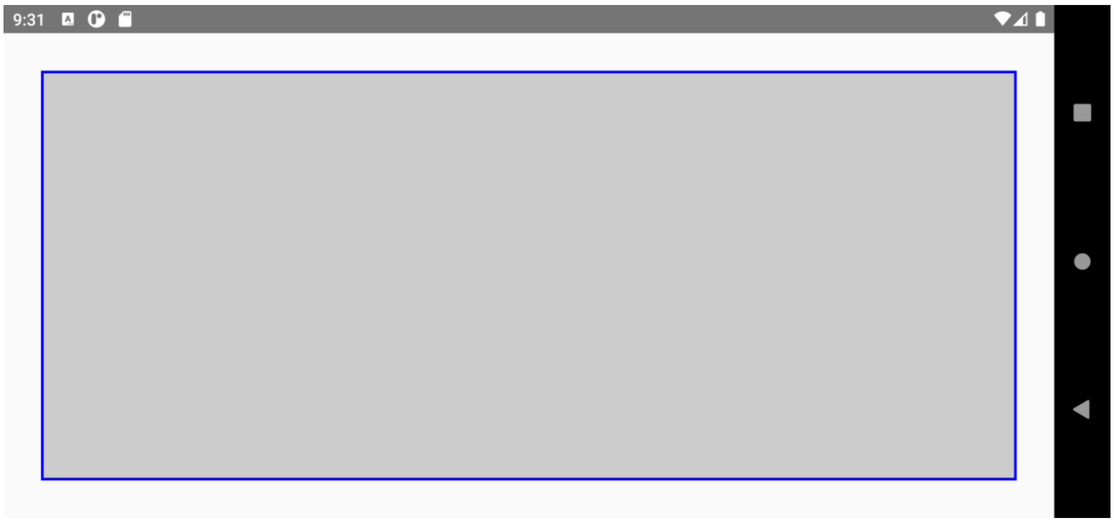


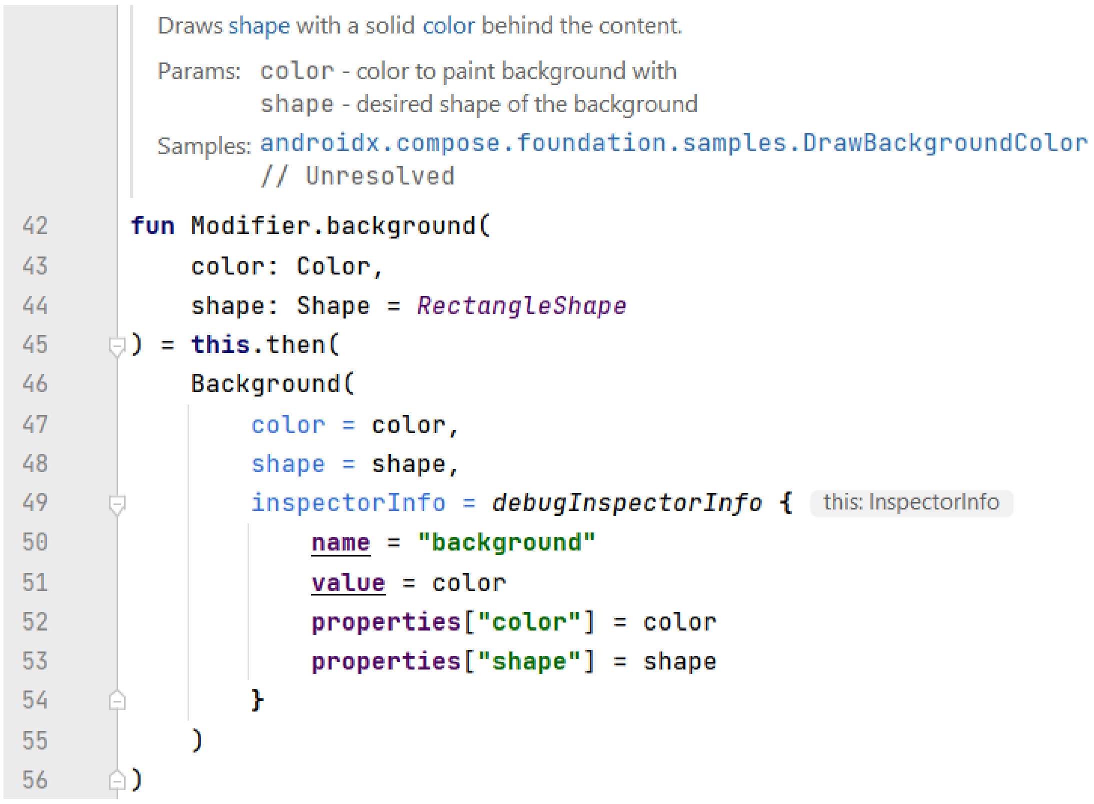
```
Interface extracted from LayoutNode to not mark the whole LayoutNode class as @PublishedApi.
@PublishedApi
internal interface ComposeUiNode {
    var measurePolicy: MeasurePolicy
    var layoutDirection: LayoutDirection
    var density: Density
    var modifier: Modifier

    Object of pre-allocated lambdas used to make use with ComposeNode allocation-less.
    companion object {
        val Constructor: () → ComposeUiNode = LayoutNode.Constructor
        val SetModifier: ComposeUiNode.(Modifier) → Unit = { this.modifier = it }
        val SetDensity: ComposeUiNode.(Density) → Unit = { this.density = it }
        val SetMeasurePolicy: ComposeUiNode.(MeasurePolicy) → Unit =
            { this.measurePolicy = it }
        val SetLayoutDirection: ComposeUiNode.(LayoutDirection) → Unit =
            { this.layoutDirection = it }
    }
}
```





```
public fun ComponentActivity.setContent(
    parent: CompositionContext? = null,
    content: @Composable () → Unit
) {
    val existingComposeView = window.decorView
        .findViewById<ViewGroup>(android.R.id.content)
        .getChildAt( index: 0) as? ComposeView

    if (existingComposeView ≠ null) with(existingComposeView) {  this: ComposeView
        setParentCompositionContext(parent)
        setContent(content)
    } else ComposeView( context: this).apply {  this: ComposeView
        // Set content and parent **before** setContentView
        // to have ComposeView create the composition on attach
        setParentCompositionContext(parent)
        setContent(content)
        // Set the view tree owners before setting the content view so that the inflation process
        // and attach listeners will see them already present
        setOwners()
        setContentView( view: this, DefaultActivityContentLayoutParams)
    }
}
```



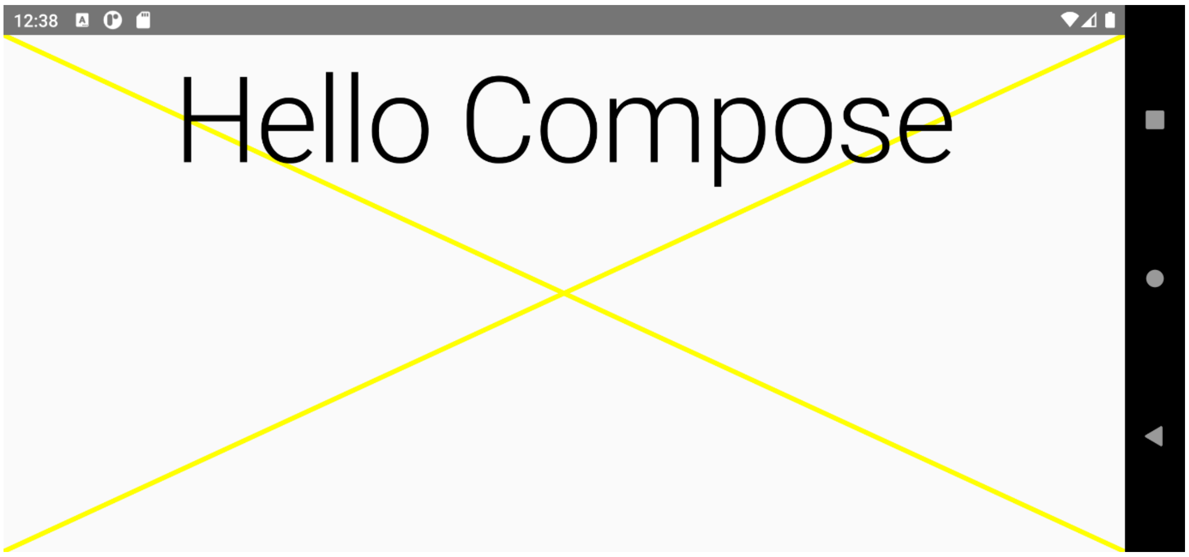


Draws shape with a solid color behind the content.

Params: color - color to paint background with
shape - desired shape of the background

Samples: androidx.compose.foundation.samples.DrawBackgroundColor
// Unresolved

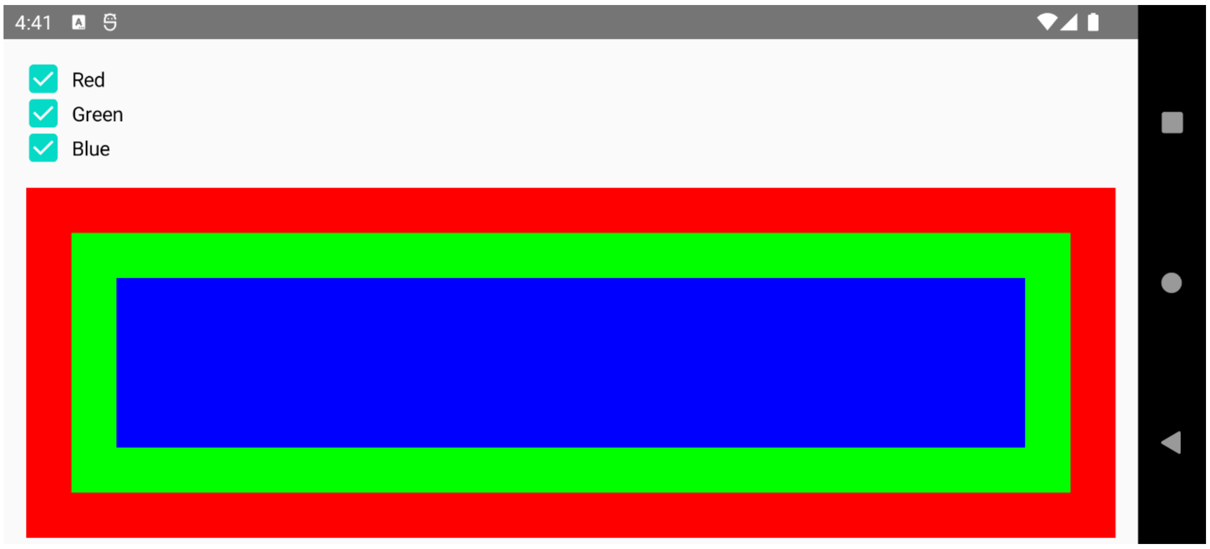
```
fun Modifier.background(
    color: Color,
    shape: Shape = RectangleShape
) = this.then(
    Background(
        color = color,
        shape = shape,
        inspectorInfo = debugInspectorInfo {  this: InspectorInfo
            name = "background"
            value = color
            properties["color"] = color
            properties["shape"] = shape
        }
    )
)
```

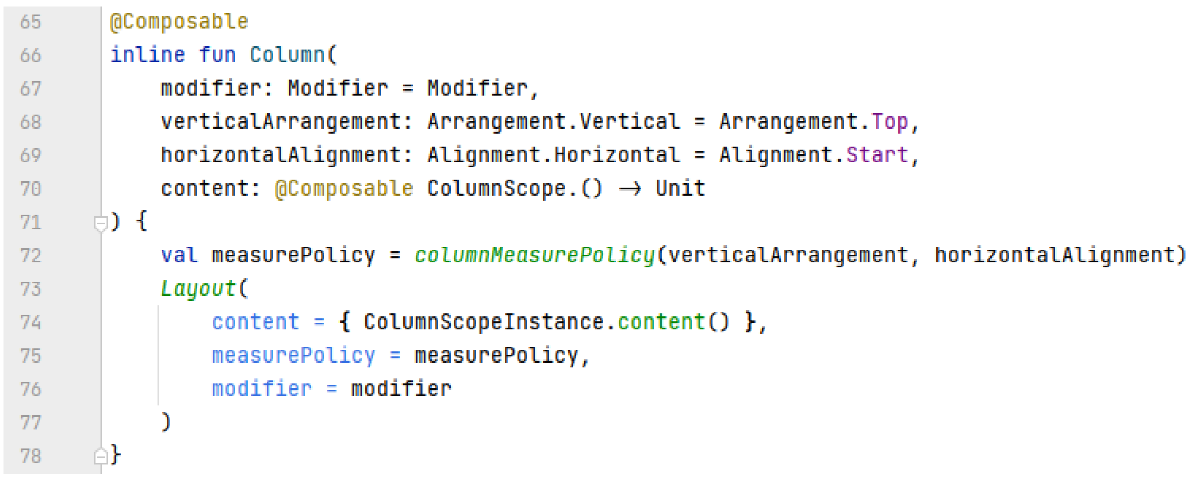


12:38

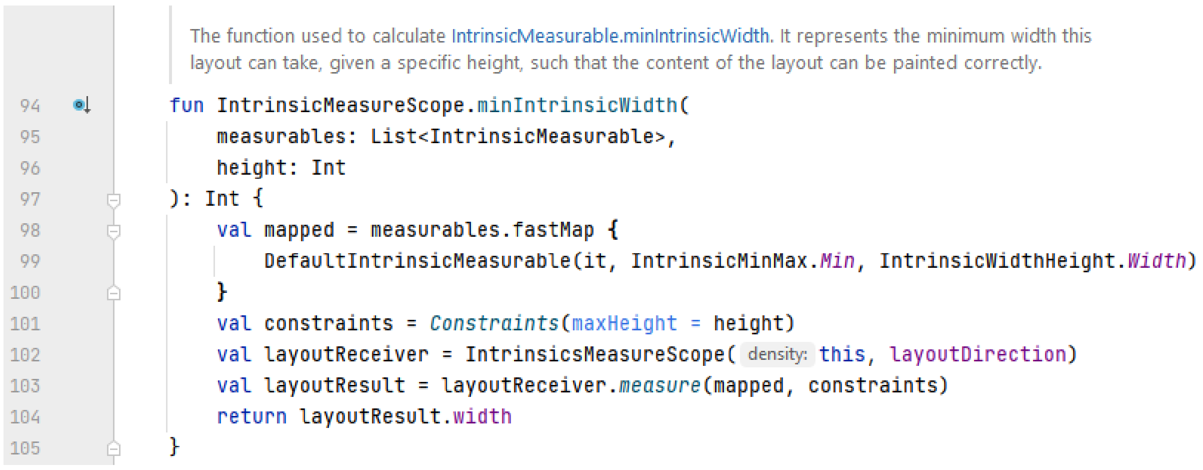
Hello Compose

Chapter 04: Laying Out UI Elements



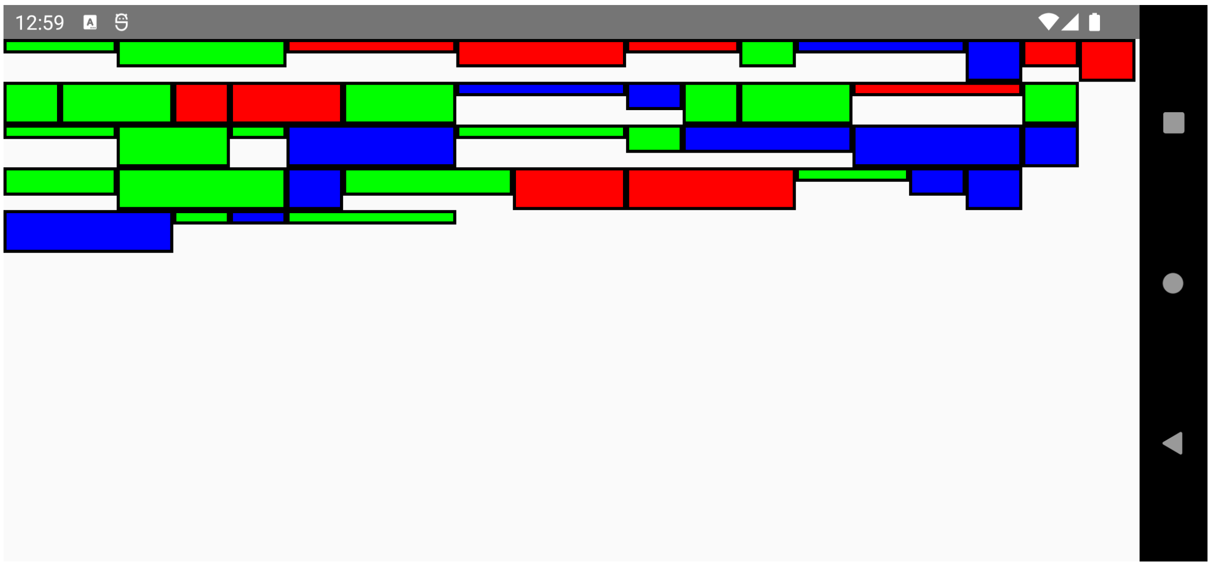


```kotlin
@Composable
inline fun Column(
    modifier: Modifier = Modifier,
    verticalArrangement: Arrangement.Vertical = Arrangement.Top,
    horizontalAlignment: Alignment.Horizontal = Alignment.Start,
    content: @Composable ColumnScope.() → Unit
) {
    val measurePolicy = columnMeasurePolicy(verticalArrangement, horizontalAlignment)
    Layout(
        content = { ColumnScopeInstance.content() },
        measurePolicy = measurePolicy,
        modifier = modifier
    )
}
```

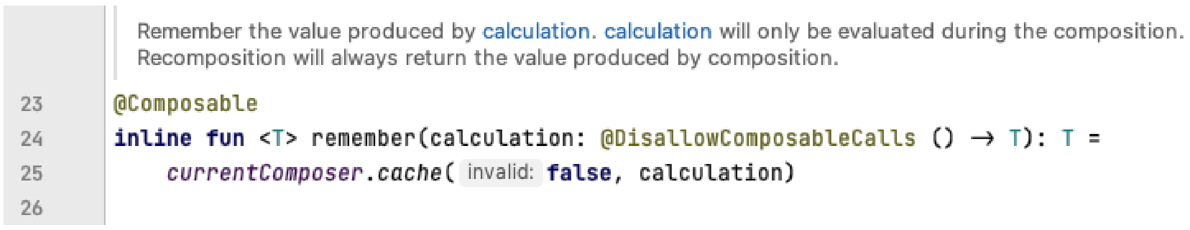


The function used to calculate IntrinsicMeasurable.minIntrinsicWidth. It represents the minimum width this layout can take, given a specific height, such that the content of the layout can be painted correctly.

```kotlin
fun IntrinsicMeasureScope.minIntrinsicWidth(
    measurables: List<IntrinsicMeasurable>,
    height: Int
): Int {
    val mapped = measurables.fastMap {
        DefaultIntrinsicMeasurable(it, IntrinsicMinMax.Min, IntrinsicWidthHeight.Width)
    }
    val constraints = Constraints(maxHeight = height)
    val layoutReceiver = IntrinsicsMeasureScope( density: this, layoutDirection)
    val layoutResult = layoutReceiver.measure(mapped, constraints)
    return layoutResult.width
}
```

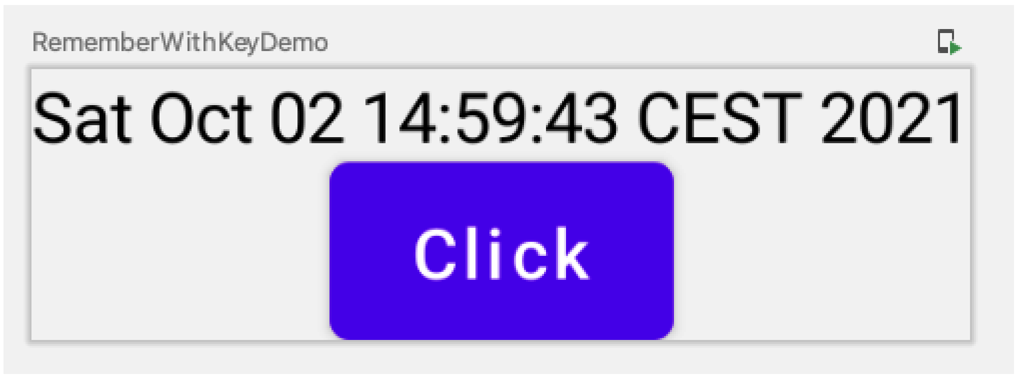

12:59

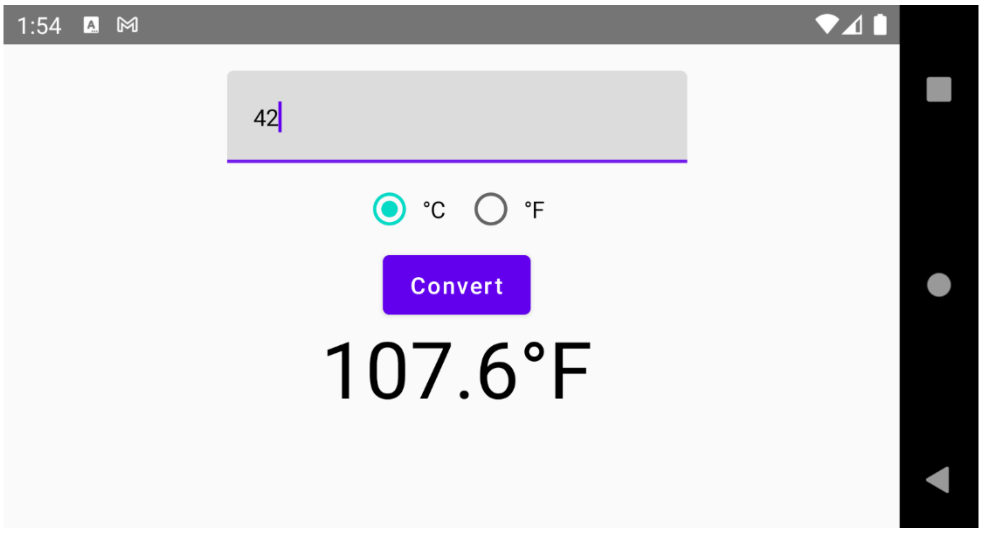
Chapter 05: Managing the State of Your Composable Functions

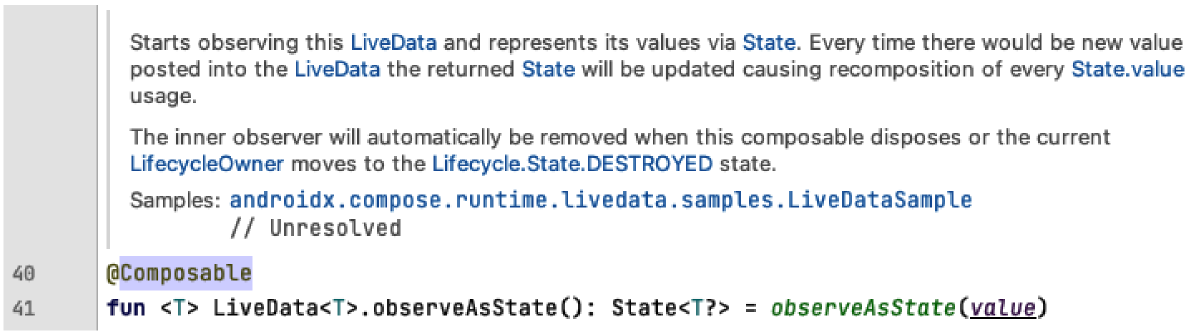


Remember the value produced by calculation. calculation will only be evaluated during the composition. Recomposition will always return the value produced by composition.

```
@Composable
inline fun <T> remember(calculation: @DisallowComposableCalls () → T): T =
    currentComposer.cache( invalid: false, calculation)

```





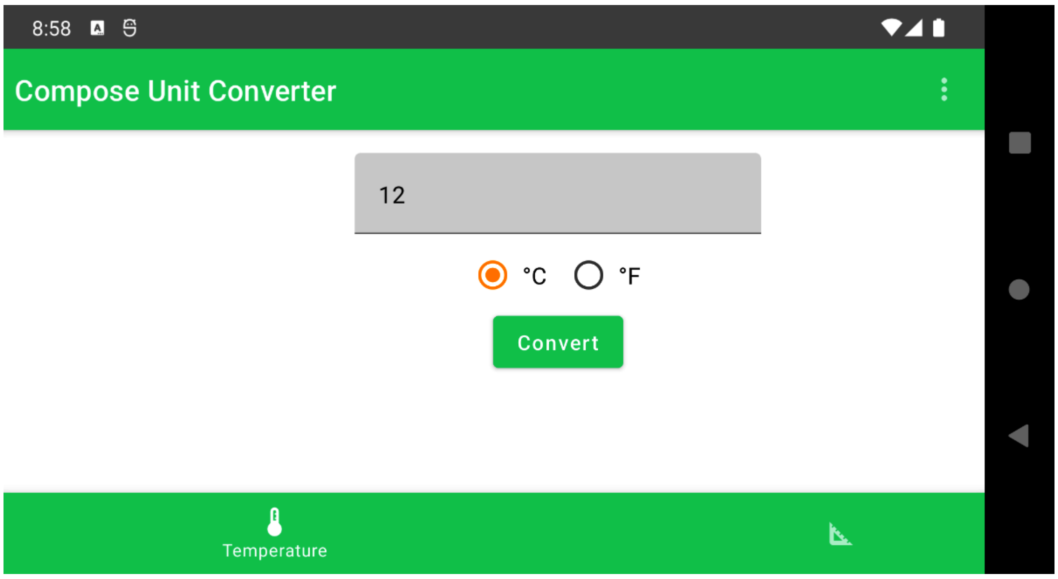

Starts observing this LiveData and represents its values via State. Every time there would be new value posted into the LiveData the returned State will be updated causing recomposition of every State.value usage.

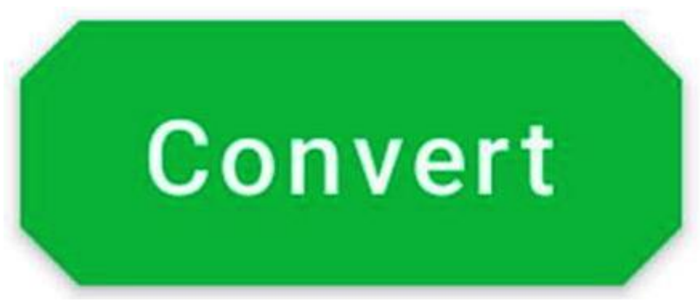
The inner observer will automatically be removed when this composable disposes or the current LifecycleOwner moves to the Lifecycle.State.DESTROYED state.

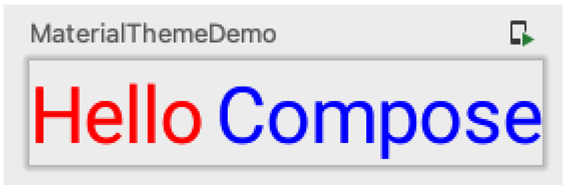
Samples: `androidx.compose.runtime.livedata.samples.LiveDataSample`
```
        // Unresolved
```

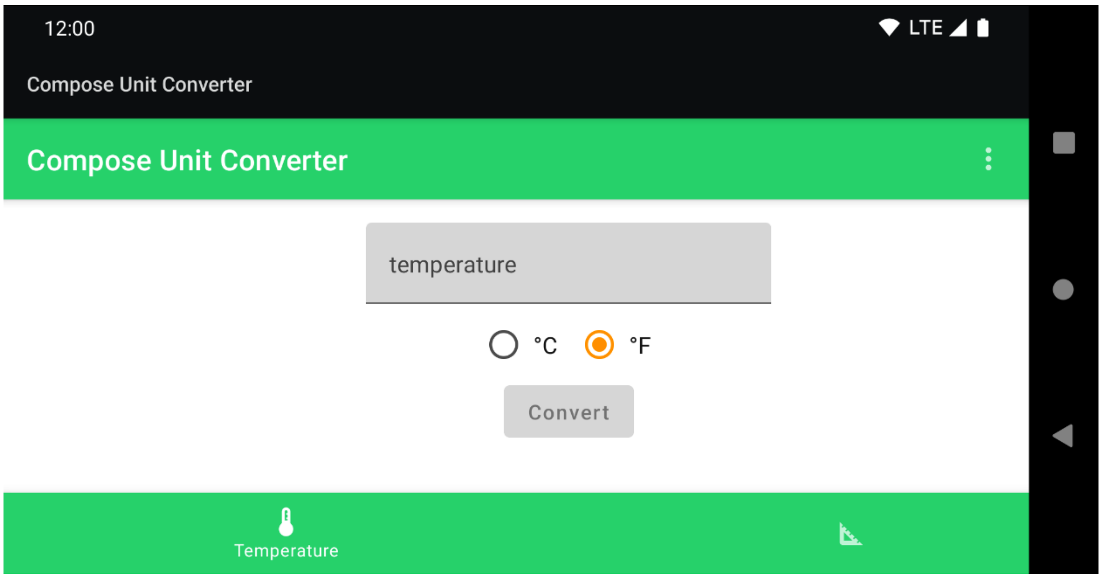
```
@Composable
fun <T> LiveData<T>.observeAsState(): State<T?> = observeAsState(value)
```

Chapter 06: Putting Pieces Together

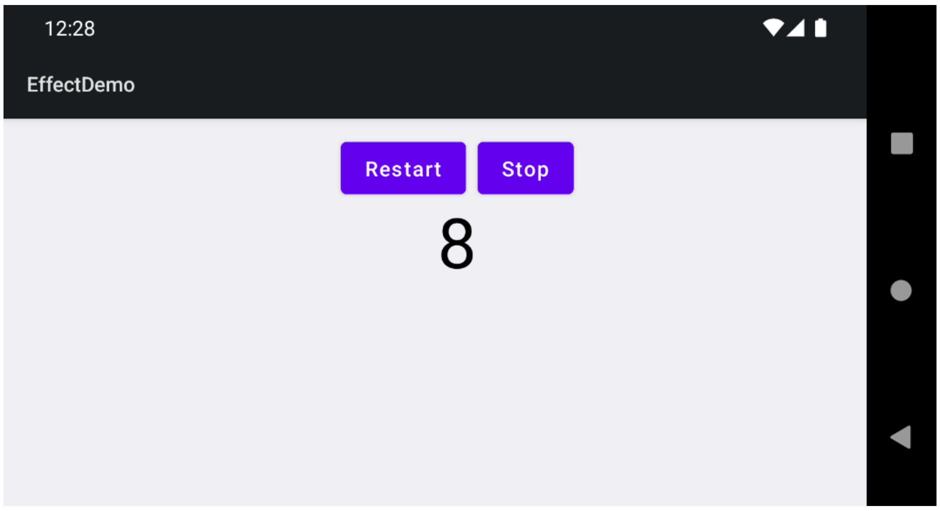




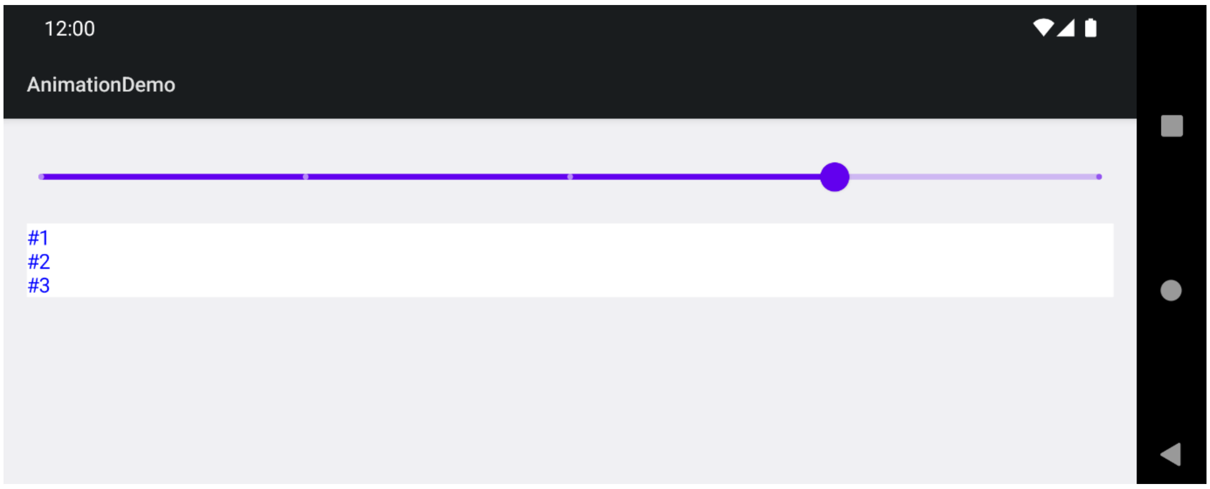


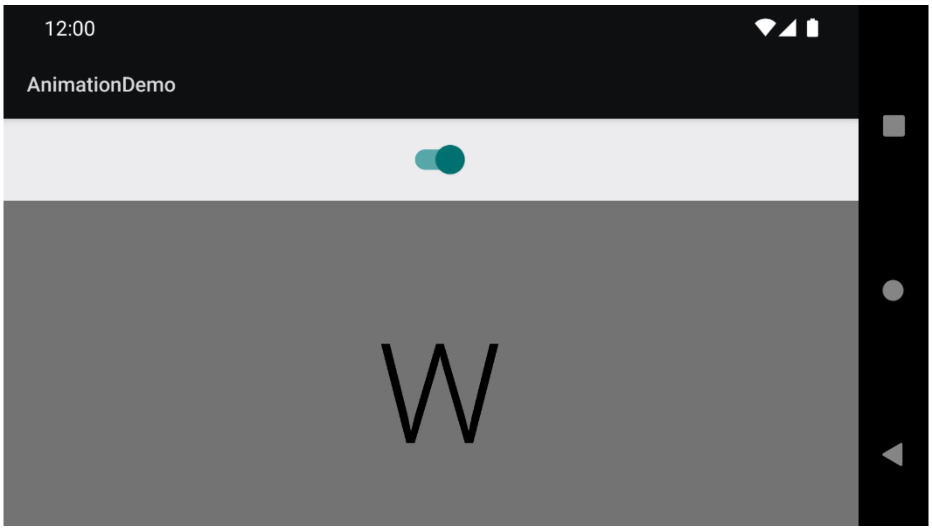

12:00
LTE
Compose Unit Converter
Compose Unit Converter
temperature
°C
°F
Convert
Temperature

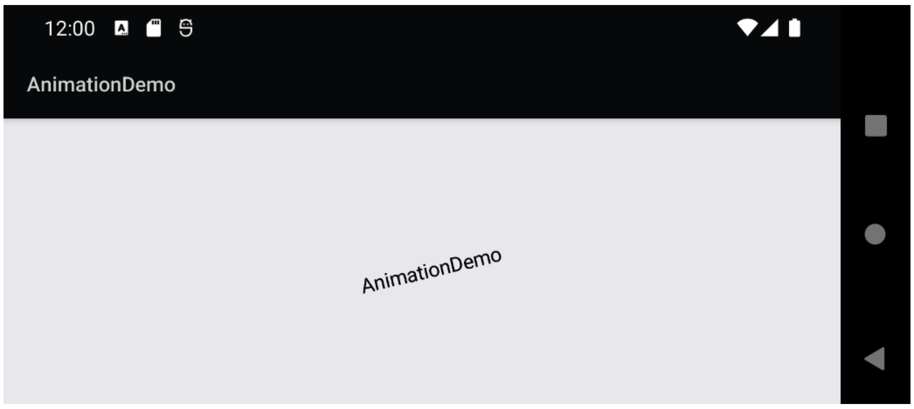
Chapter 07: Tips, Tricks, and Best Practices

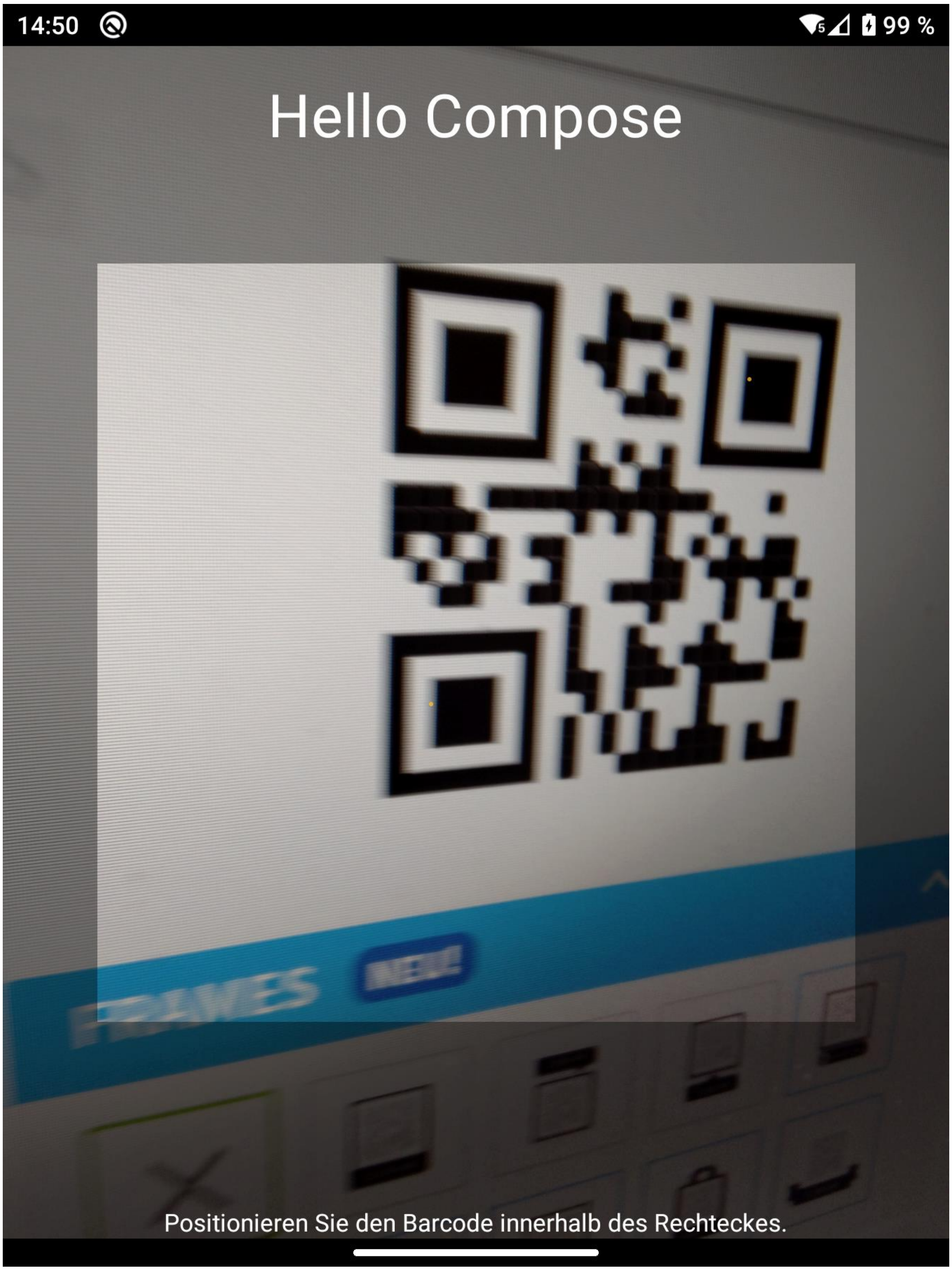


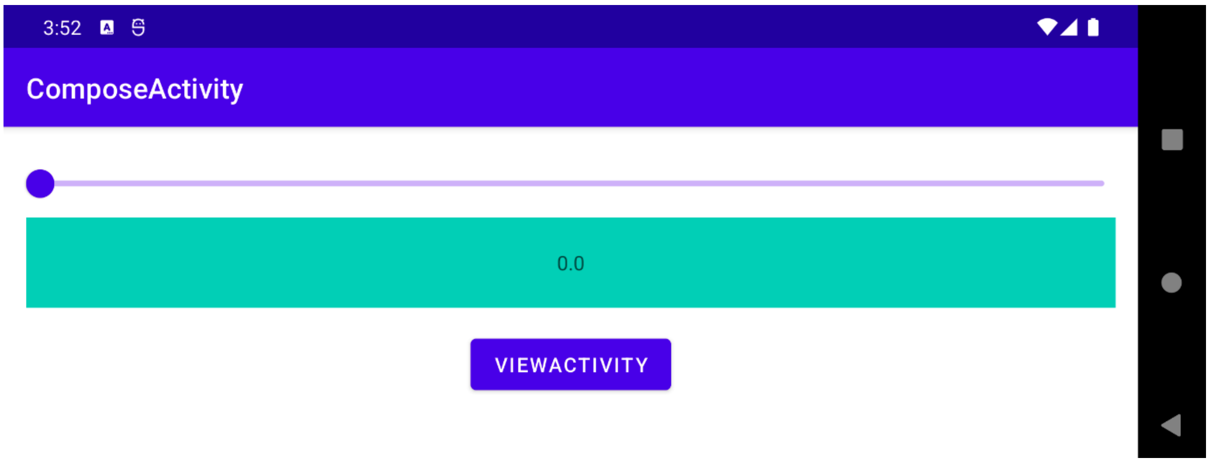
Chapter 08: Working with Animations

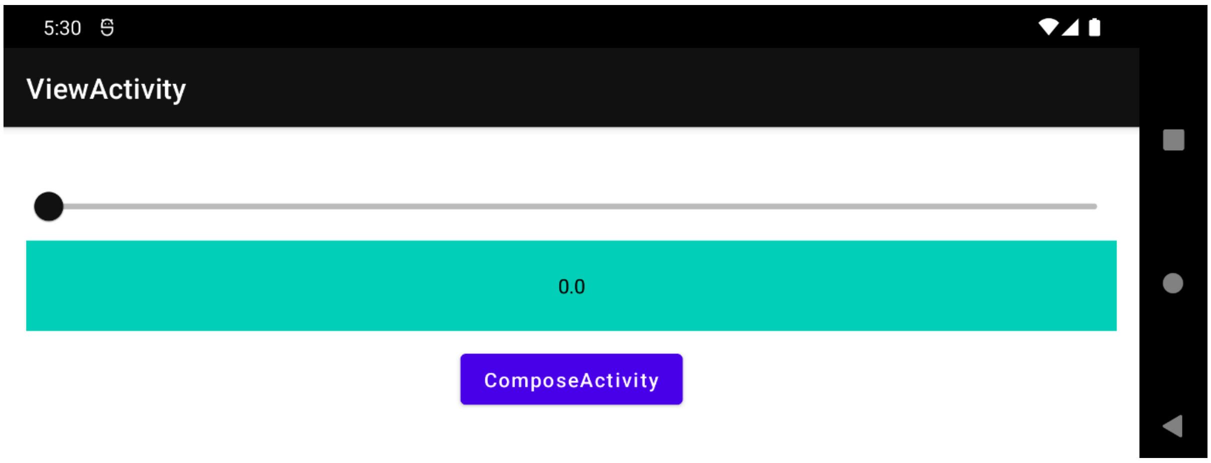




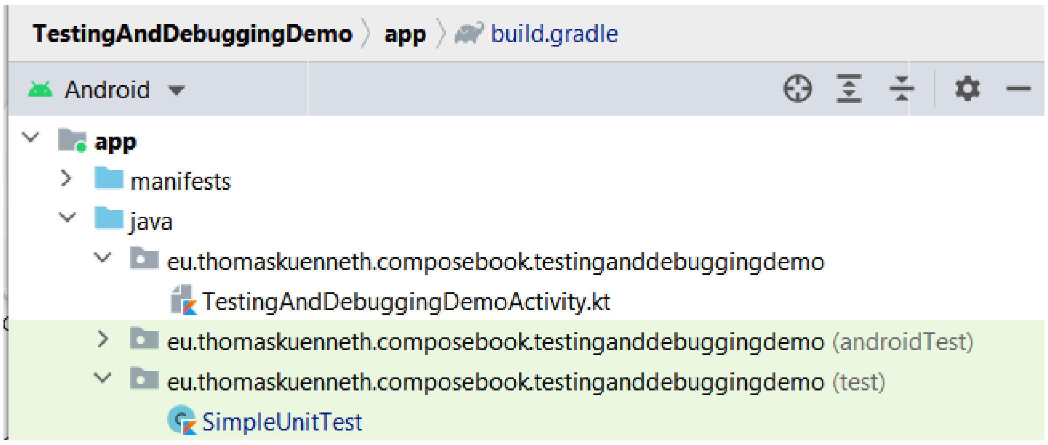



14:50
99 %
Hello Compose
Positionieren Sie den Barcode innerhalb des Rechteckes.


3:52
ComposeActivity
0.0
VIEWACTIVITY


5:30
ViewActivity
0.0
ComposeActivity

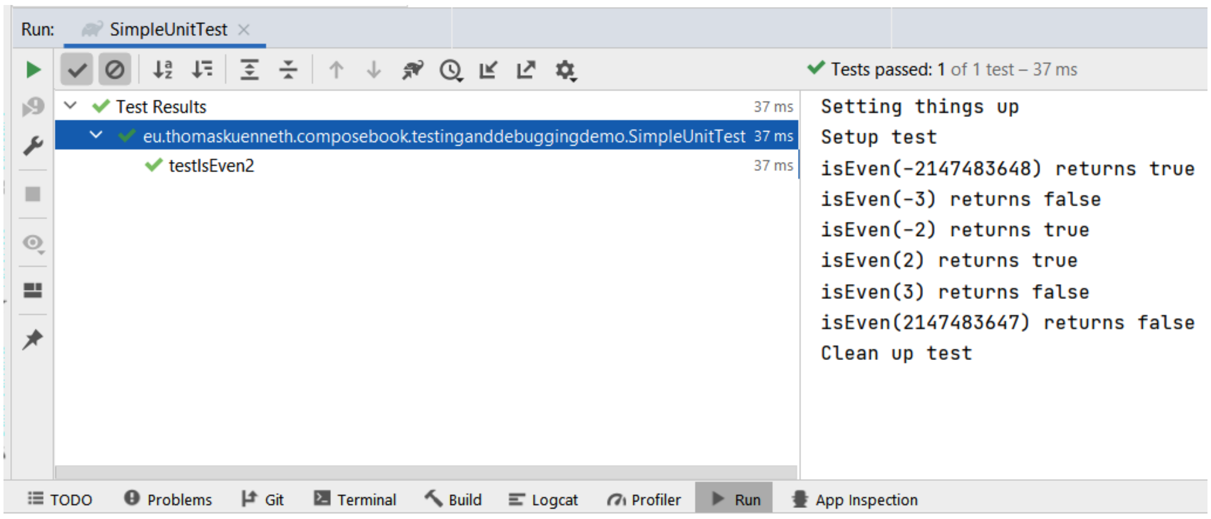
Chapter 10: Testing and Debugging Compose Apps



TestingAndDebuggingDemo 〉 app 〉 build.gradle

Android

- app
  - manifests
  - java
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo
      - TestingAndDebuggingDemoActivity.kt
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo (androidTest)
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo (test)
      - SimpleUnitTest

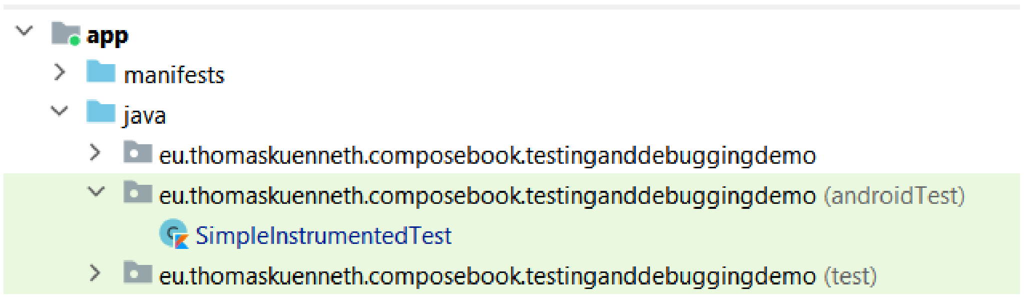


Run: SimpleUnitTest

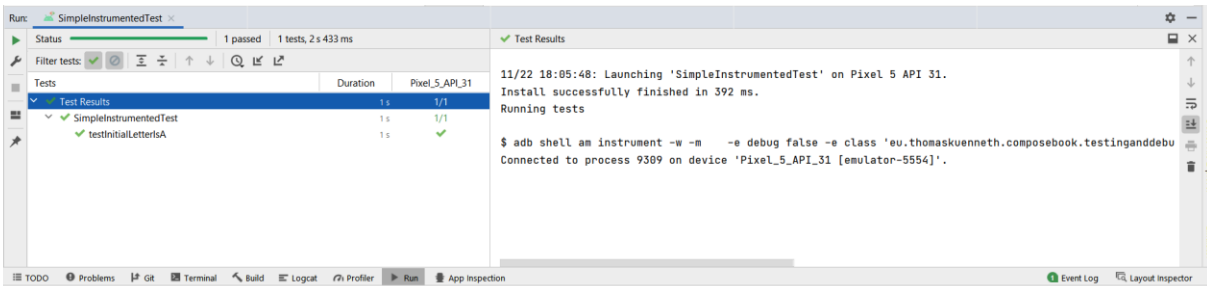
Tests passed: 1 of 1 test – 37 ms

```
Test Results                                                                37 ms
  eu.thomaskuenneth.composebook.testinganddebuggingdemo.SimpleUnitTest      37 ms
    testIsEven2                                                             37 ms
```

```
Setting things up
Setup test
isEven(-2147483648) returns true
isEven(-3) returns false
isEven(-2) returns true
isEven(2) returns true
isEven(3) returns false
isEven(2147483647) returns false
Clean up test
```



- app
  - manifests
  - java
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo (androidTest)
      - SimpleInstrumentedTest
    - eu.thomaskuenneth.composebook.testinganddebuggingdemo (test)

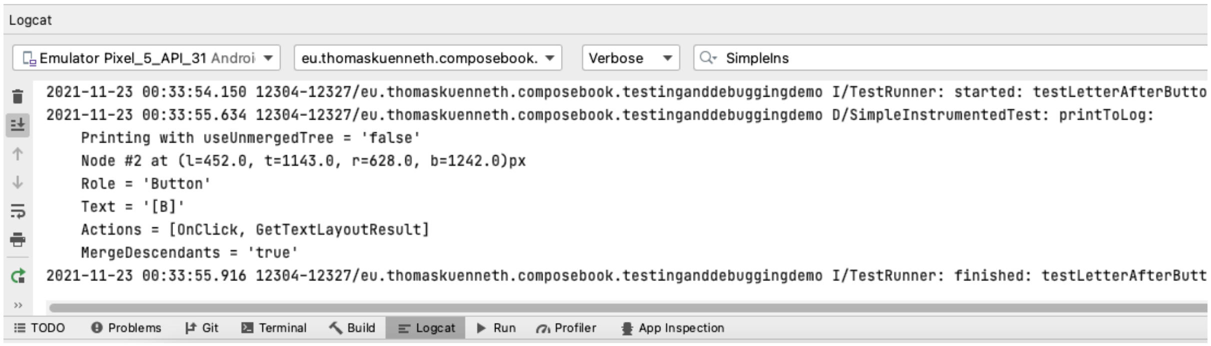


Tests	Duration	Pixel_5_API_31
Test Results	1 s	1/1
SimpleInstrumentedTest	1 s	1/1
testInitialLetterIsA	1 s	✓

Status 1 passed 1 tests, 2 s 433 ms

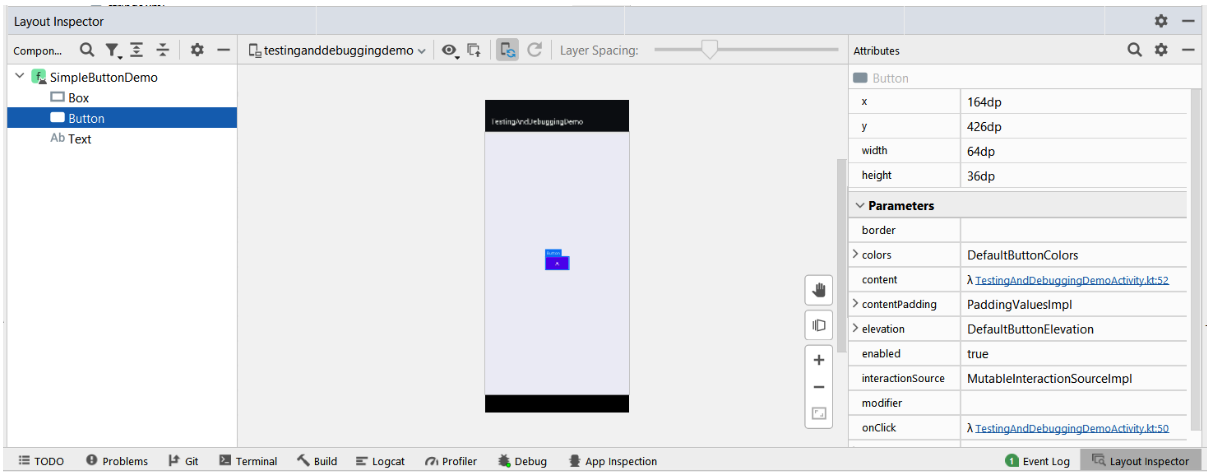
Test Results

```
11/22 18:05:48: Launching 'SimpleInstrumentedTest' on Pixel 5 API 31.
Install successfully finished in 392 ms.
Running tests

$ adb shell am instrument -w -m    -e debug false -e class 'eu.thomaskuenneth.composebook.testinganddebu
Connected to process 9309 on device 'Pixel_5_API_31 [emulator-5554]'.
```



Emulator Pixel_5_API_31 Androi | eu.thomaskuenneth.composebook. | Verbose | SimpleIns

```
2021-11-23 00:33:54.150 12304-12327/eu.thomaskuenneth.composebook.testinganddebuggingdemo I/TestRunner: started: testLetterAfterButto
2021-11-23 00:33:55.634 12304-12327/eu.thomaskuenneth.composebook.testinganddebuggingdemo D/SimpleInstrumentedTest: printToLog:
    Printing with useUnmergedTree = 'false'
    Node #2 at (l=452.0, t=1143.0, r=628.0, b=1242.0)px
    Role = 'Button'
    Text = '[B]'
    Actions = [OnClick, GetTextLayoutResult]
    MergeDescendants = 'true'
2021-11-23 00:33:55.916 12304-12327/eu.thomaskuenneth.composebook.testinganddebuggingdemo I/TestRunner: finished: testLetterAfterButt
```

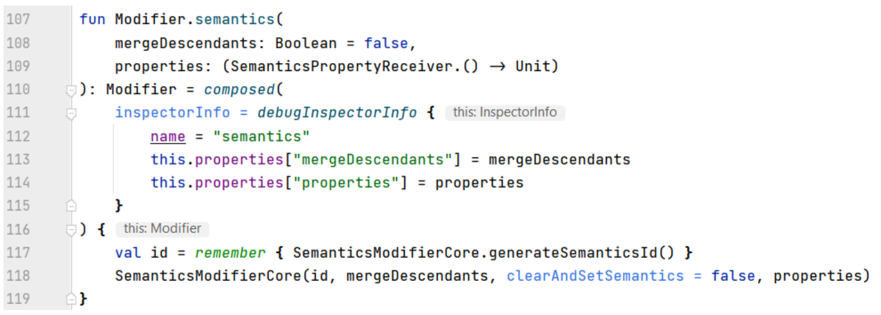


Button

x	164dp
y	426dp
width	64dp
height	36dp

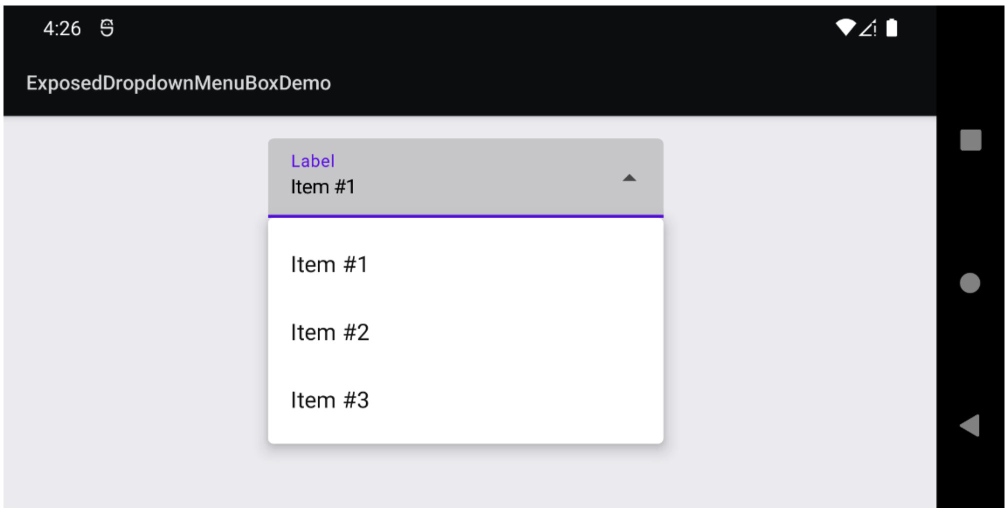
Parameters

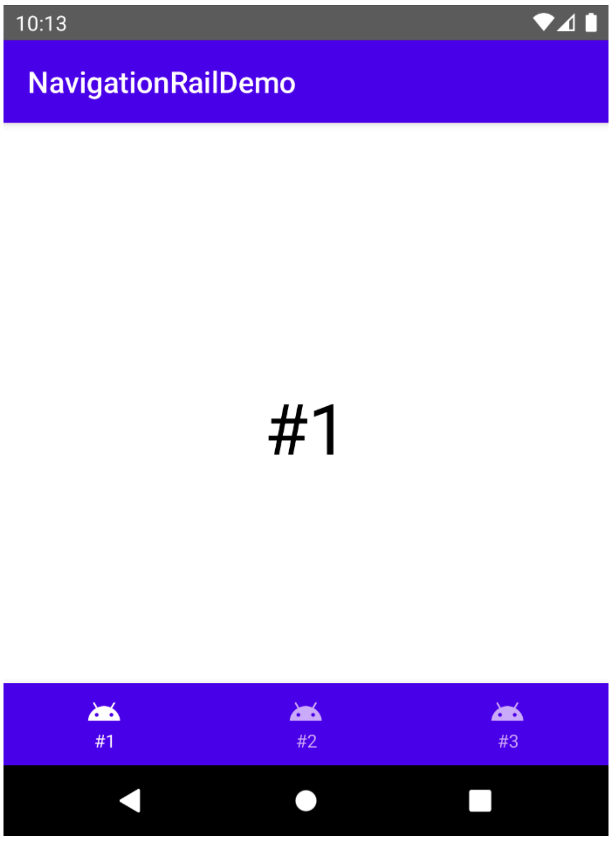
border	
colors	DefaultButtonColors
content	λ TestingAndDebuggingDemoActivity.kt:52
contentPadding	PaddingValuesImpl
elevation	DefaultButtonElevation
enabled	true
interactionSource	MutableInteractionSourceImpl
modifier	
onClick	λ TestingAndDebuggingDemoActivity.kt:50

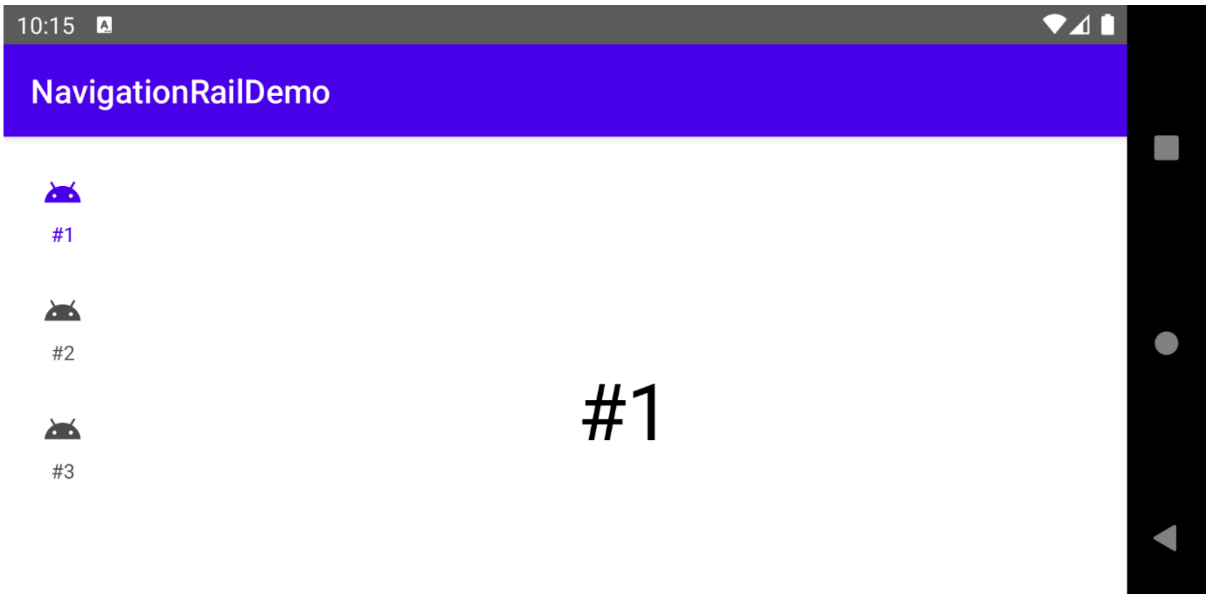


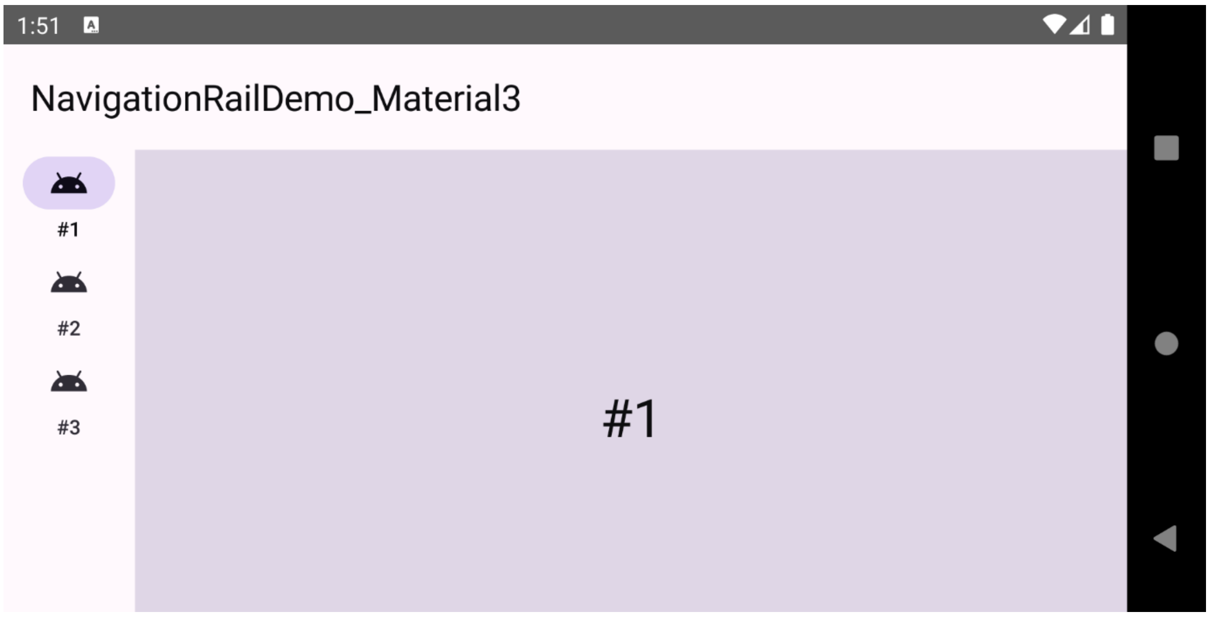
```
fun Modifier.semantics(
    mergeDescendants: Boolean = false,
    properties: (SemanticsPropertyReceiver.() -> Unit)
): Modifier = composed(
    inspectorInfo = debugInspectorInfo {  this: InspectorInfo
        name = "semantics"
        this.properties["mergeDescendants"] = mergeDescendants
        this.properties["properties"] = properties
    }
) {  this: Modifier
    val id = remember { SemanticsModifierCore.generateSemanticsId() }
    SemanticsModifierCore(id, mergeDescendants, clearAndSetSemantics = false, properties)
}
```

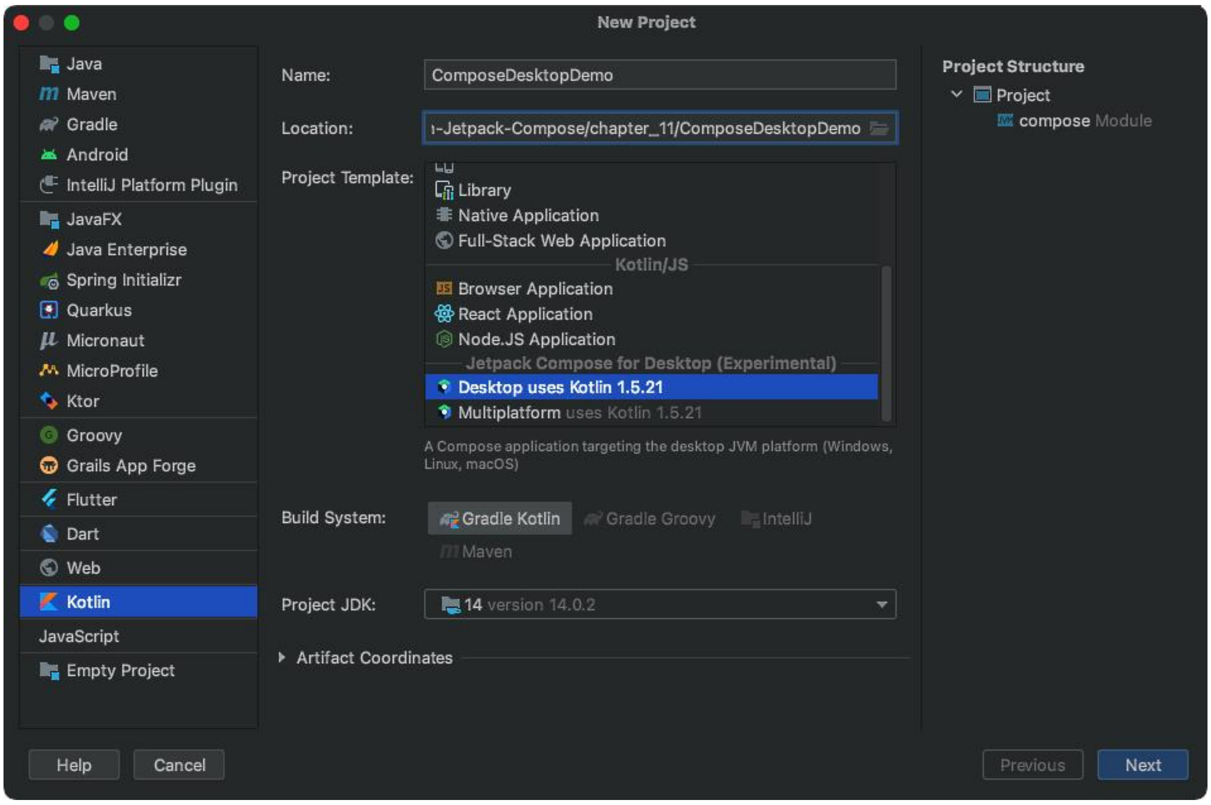
Chapter 11: Conclusion and Next Steps






10:15
NavigationRailDemo
#1
#2
#3
#1


1:51
NavigationRailDemo_Material3
#1
#2
#3
#1



Name: ComposeDesktopDemo

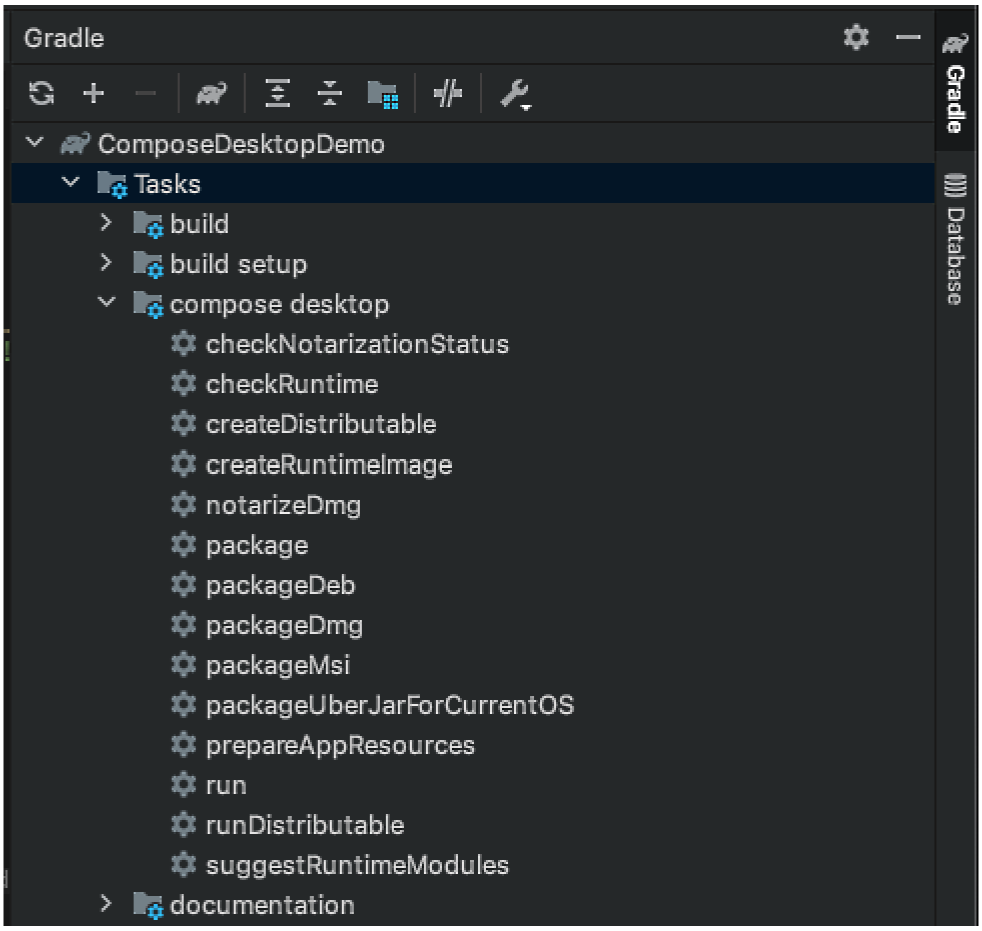
Location: ı-Jetpack-Compose/chapter_11/ComposeDesktopDemo

Project Template: Desktop uses Kotlin 1.5.21

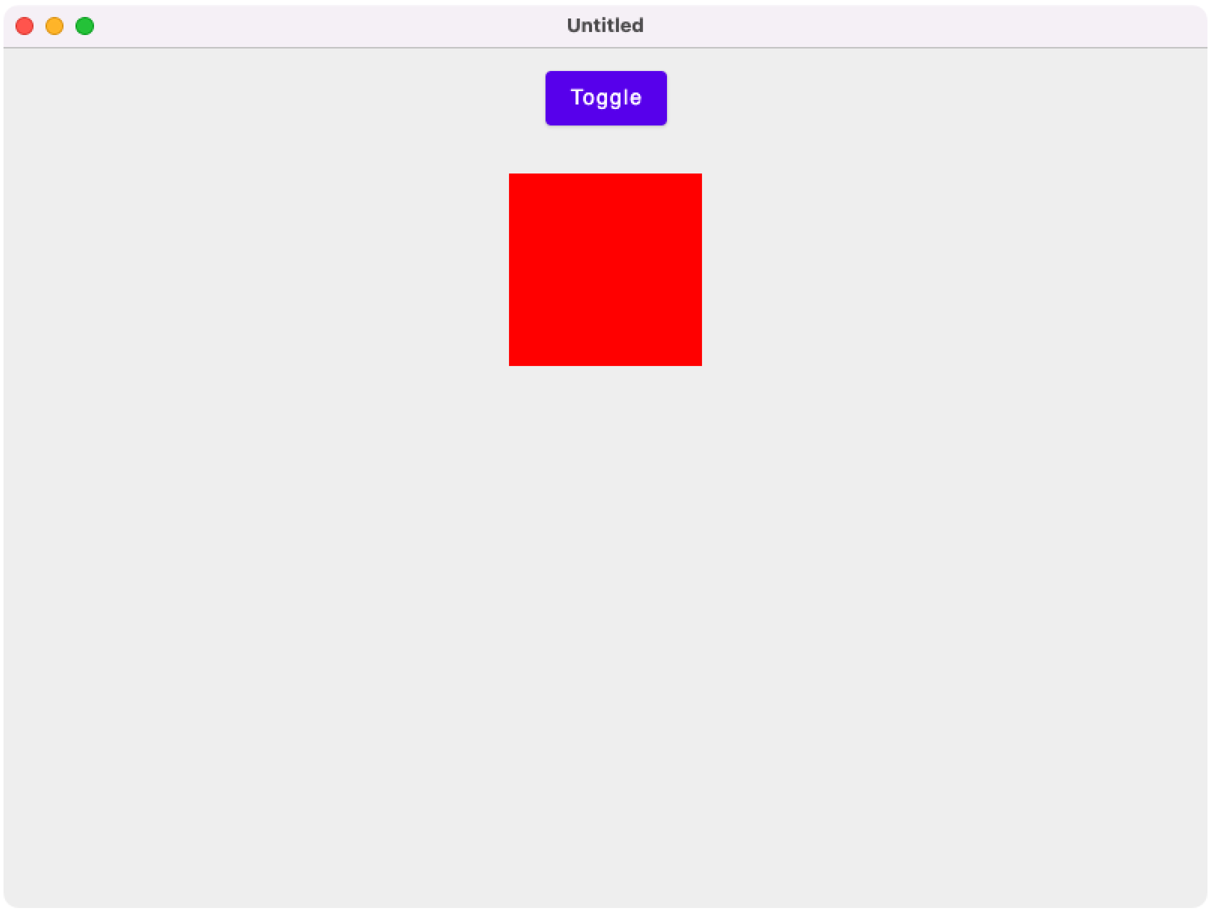
A Compose application targeting the desktop JVM platform (Windows, Linux, macOS)

Build System: Gradle Kotlin

Project JDK: 14 version 14.0.2



```
ComposeDesktopDemo
  Tasks
    build
    build setup
    compose desktop
      checkNotarizationStatus
      checkRuntime
      createDistributable
      createRuntimeImage
      notarizeDmg
      package
      packageDeb
      packageDmg
      packageMsi
      packageUberJarForCurrentOS
      prepareAppResources
      run
      runDistributable
      suggestRuntimeModules
    documentation
```


Untitled
Toggle