Hacker News Compose

Building our first declarative UI on Android

A.J. Kueterman | July 9, 2020 | *SLAC* 001

Pre-Work

MacBook setup

- Download Android Studio Canary (4.2)
- Create an AVD targeting API 29 or later
- Clone the starter repository & open it in AS 4.2

What we'll cover…

- I'll introduce Jetpack Compose and its basic concepts
- We'll discuss the patterns it follows and enforces
- We'll build an Android UI in Compose
- We'll get some real data back from the Hacker News API

and what we won't.

- We won't pull the covers back too far to get into how Compose works, but I'll have resources to explore after the session
- We aren't on the bleeding edge of compose (we're using dev10 instead of, the latest, dev14)

The *what* and the *why* of Jetpack Compose

What is Compose?

- *Jetpack Compose is a modern toolkit for building native Android UI’s*
- It leaves behind the concept of Android `View`, and relies on entirely new all-Kotlin API’s to create layouts
- Simply put, it’s a framework that offers a ***declarative*** way to build our Android apps

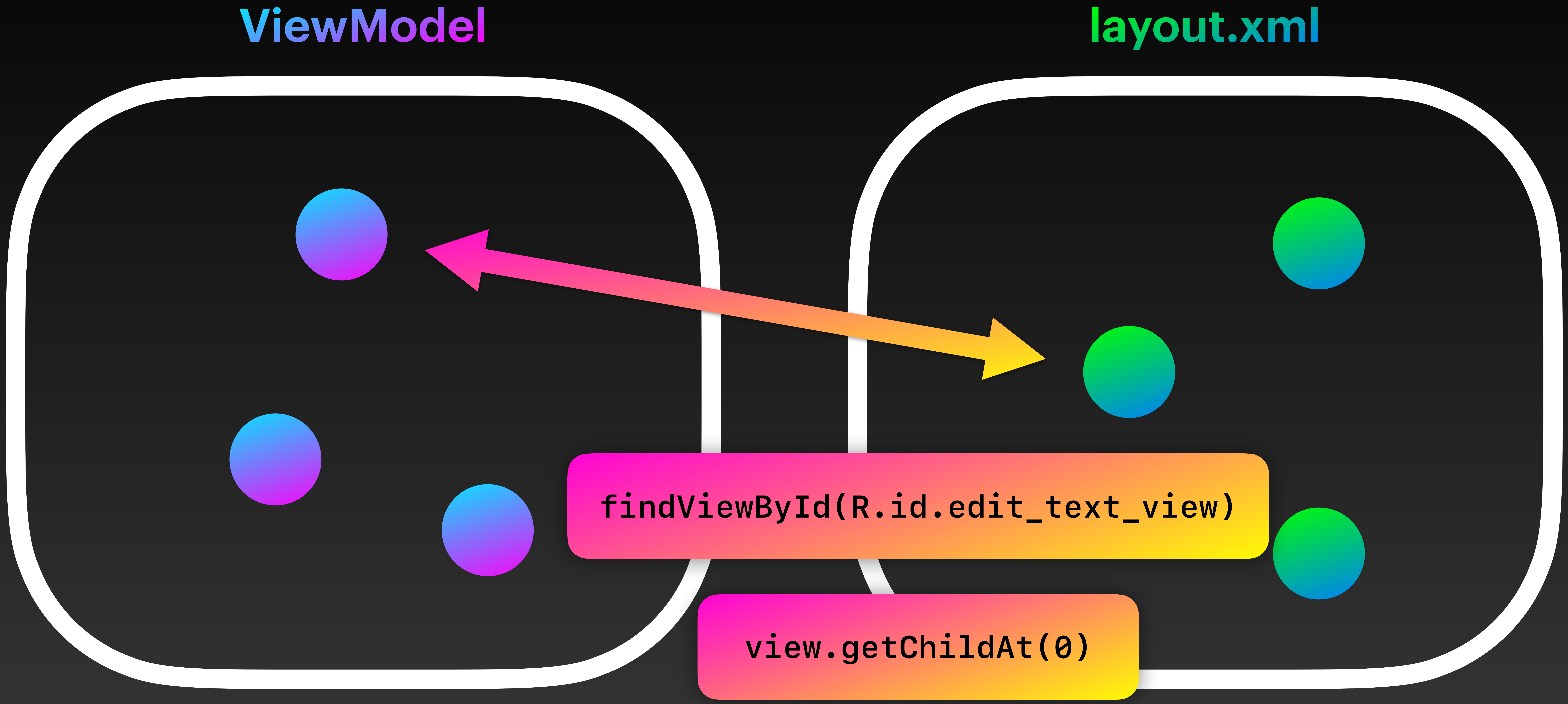
Imperative UI

```
<Button
  android:id=“@+id/button”
/>
```

```
val button = findViewById(R.id.button)
button.setText(viewModel.buttonLabel)
button.setBackgroundColor(/* color */)
button.setOnClickListener(viewModel.buttonClick)

viewModel.buttonLabel.observe(
  viewLifeCycleObserver,
  Observer { buttonLabel ->
    button.setText(buttonLabel)
  }
)
```

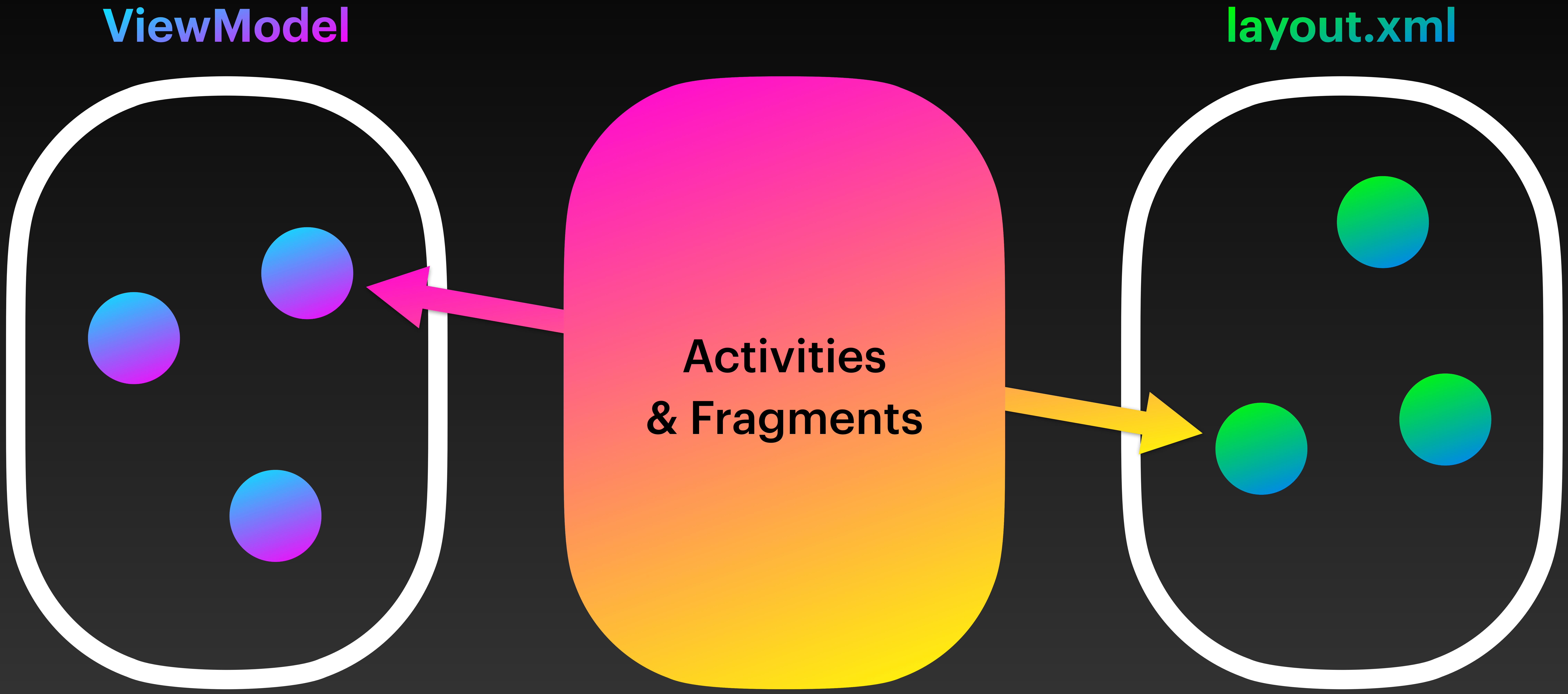
Declarative UI

```
Button(
  onClick = viewModel.buttonClick,
  backgroundColor = /* color */
) {
  Text(viewModel.buttonLabel)
}
```



```
findViewById(R.id.edit_text_view)
```

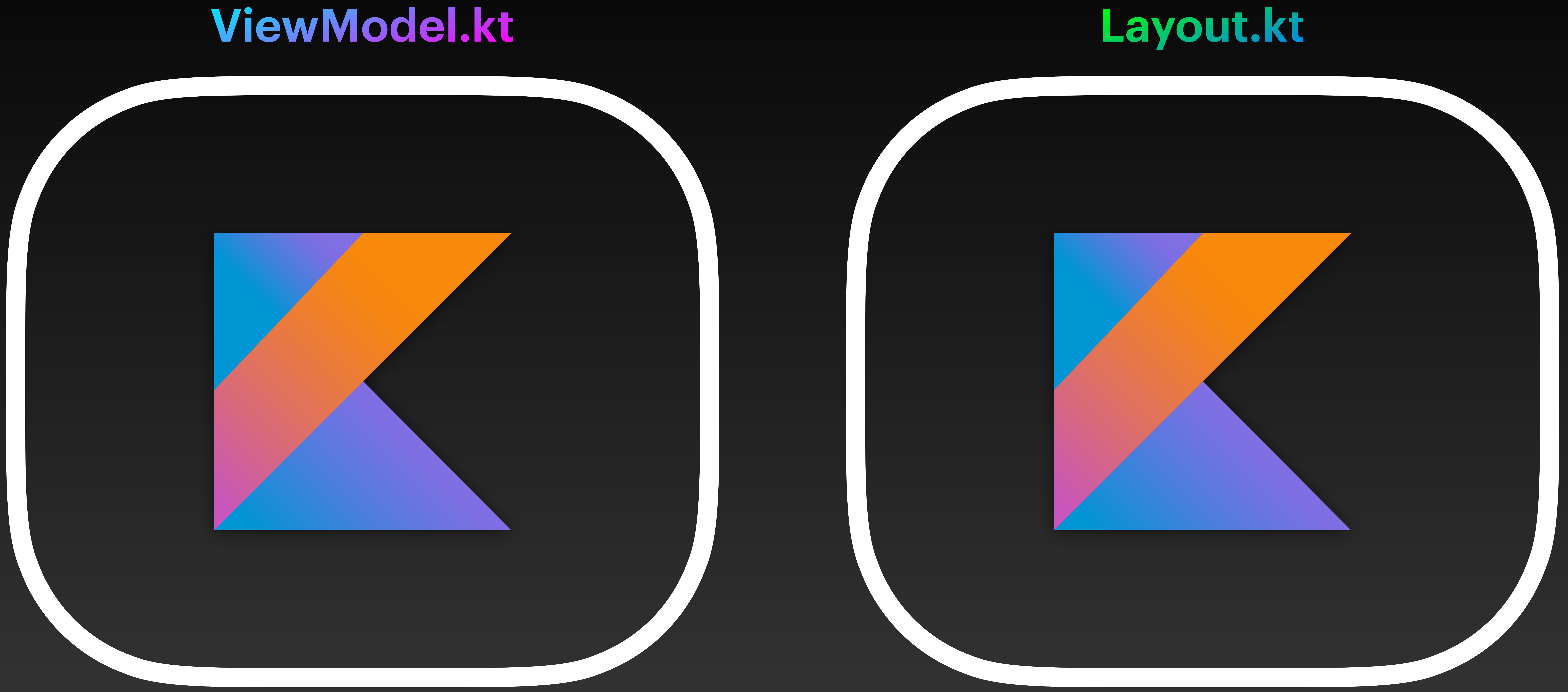
```
view.getChildAt(0)
```



ViewModel

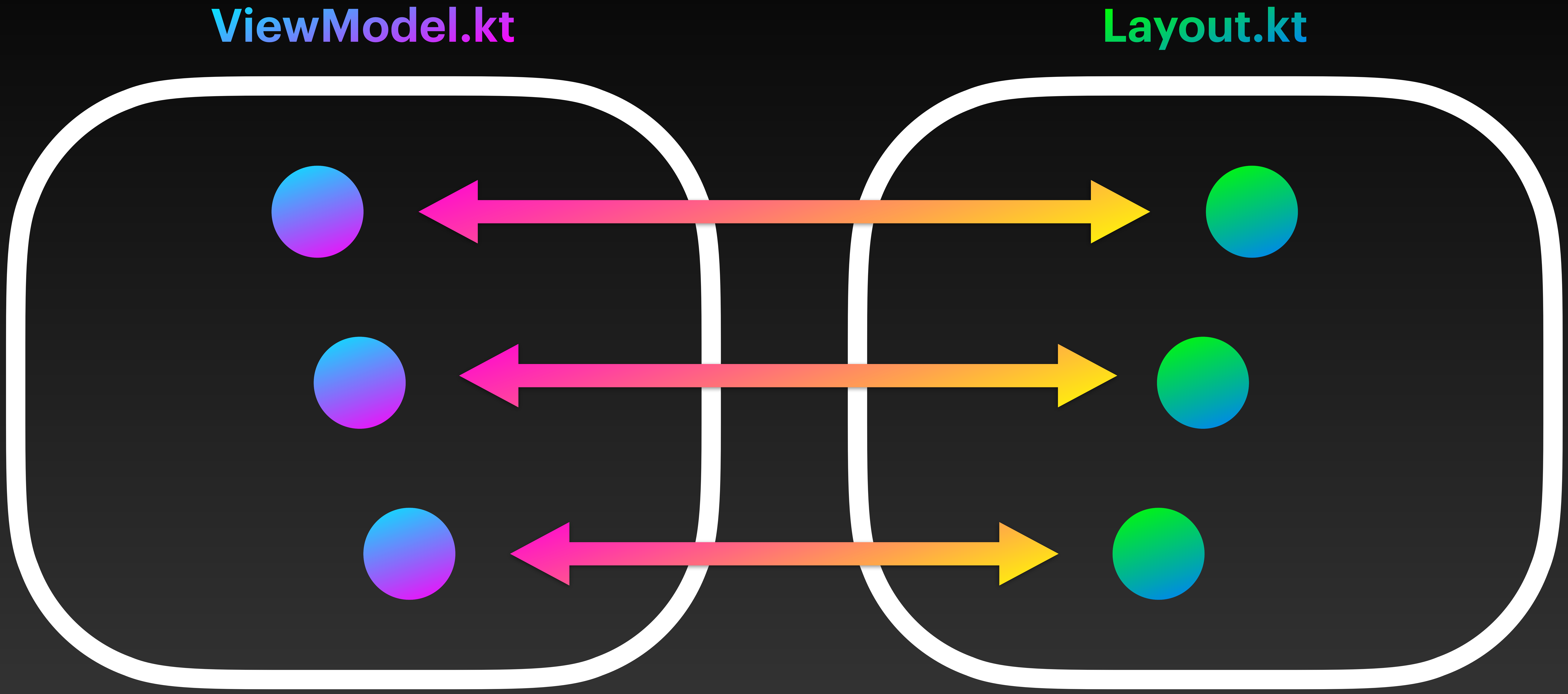
Activities
& Fragments

layout.xml



ViewModel.kt

Layout.kt



ViewModel.kt

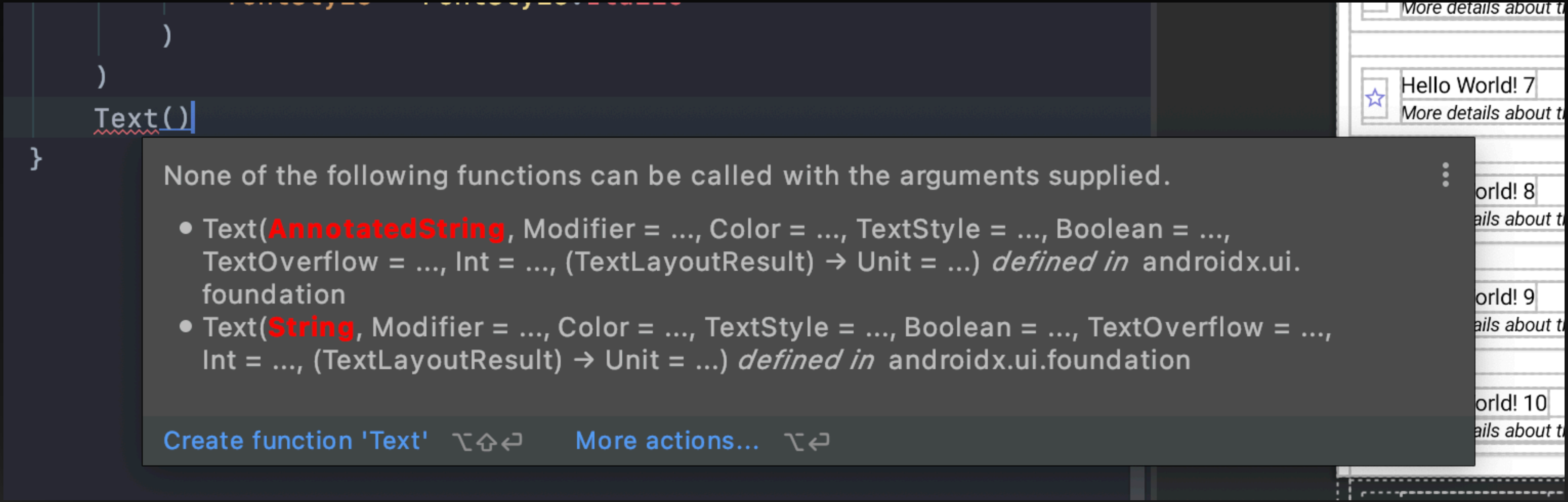
Layout.kt

Compose can’t separate your concerns for you.

```
<TextView
  android:id="@+id/greeting"
/>
```

```
val greeting = findViewById(R.id.greeting)
greeting.text = viewModel.greeting
```

```
Text(“Hello World!”)
```



```
        )
    )
    Text()
}
```

None of the following functions can be called with the arguments supplied.

- Text(AnnotatedString, Modifier = ..., Color = ..., TextStyle = ..., Boolean = ..., TextOverflow = ..., Int = ..., (TextLayoutResult) → Unit = ...) *defined in* androidx.ui.foundation
- Text(String, Modifier = ..., Color = ..., TextStyle = ..., Boolean = ..., TextOverflow = ..., Int = ..., (TextLayoutResult) → Unit = ...) *defined in* androidx.ui.foundation

Create function 'Text' ⌥⇧↩ More actions... ⌥↩

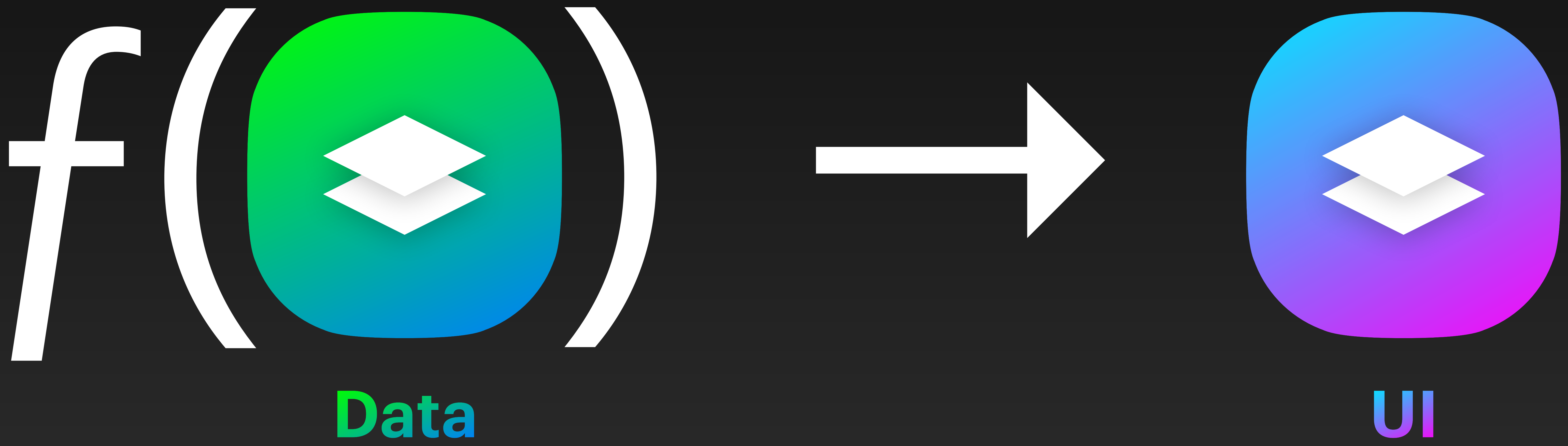
```
@Composable
fun TitleLabel() {

}
```

```kotlin
@Composable
fun TitleLabel(title: String) {


}
```

```
@Composable
fun TitleLabel(title: String) {
  Text(“Read about: $title”)
}
```

```
@Composable
fun TitleCard(title: String) {
  TitleLabel(title)
  Text(“Story Text”)
}

@Composable
fun TitleLabel(title: String) {
  Text(“Read about: $title”)
}
```

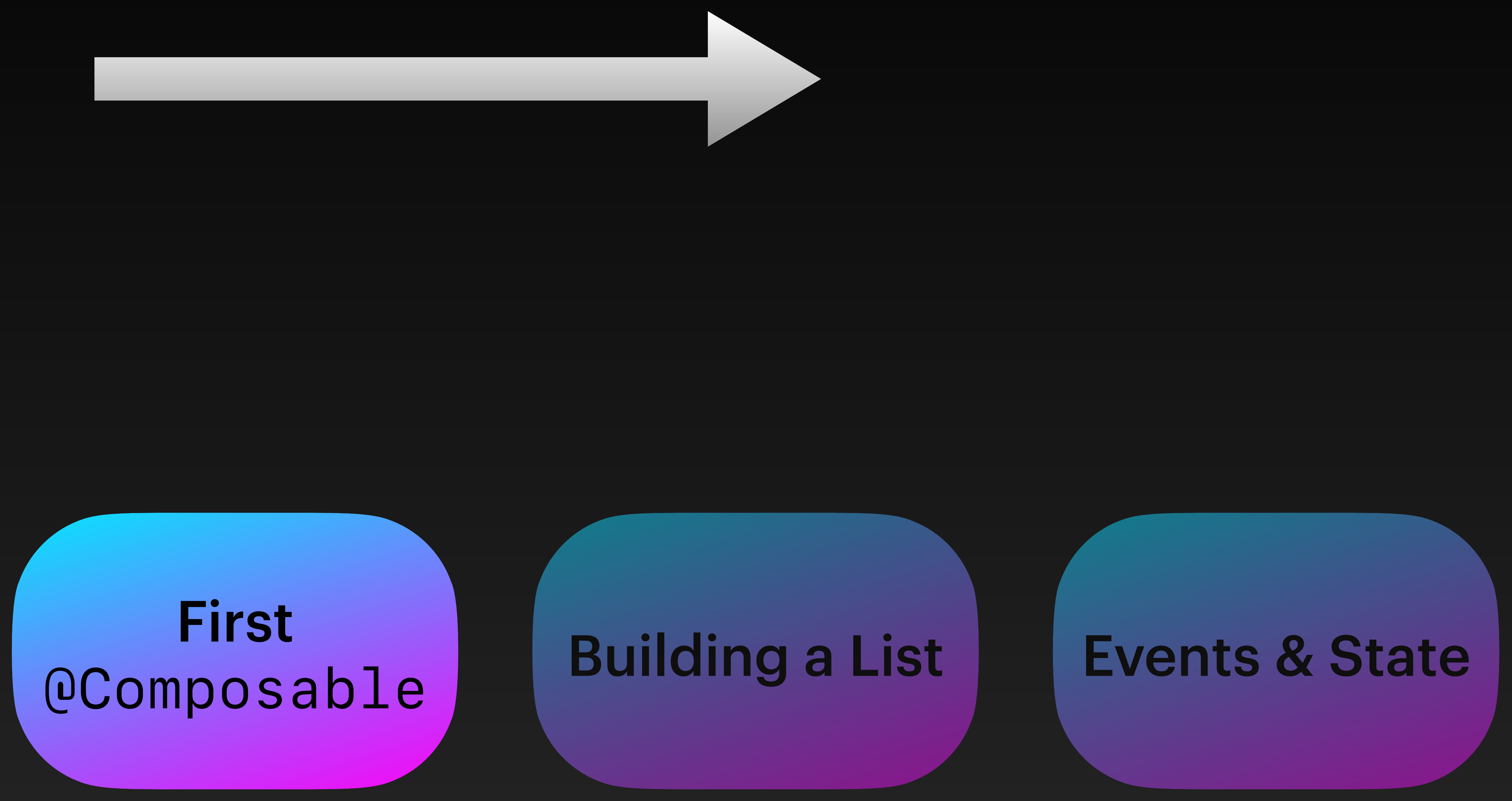
Functions all the way down



Data

UI

Flow



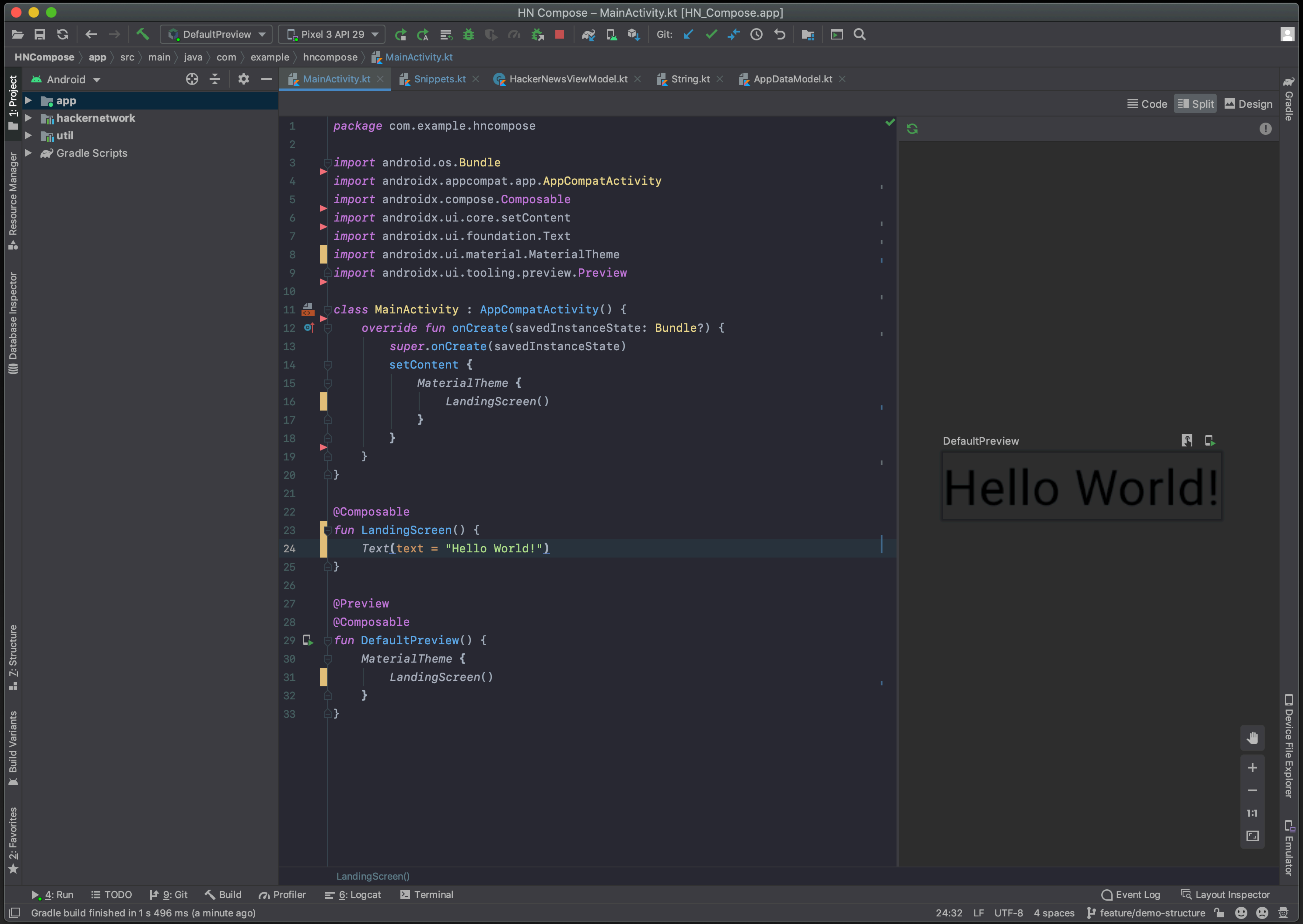
First `@Composable`

Building a List

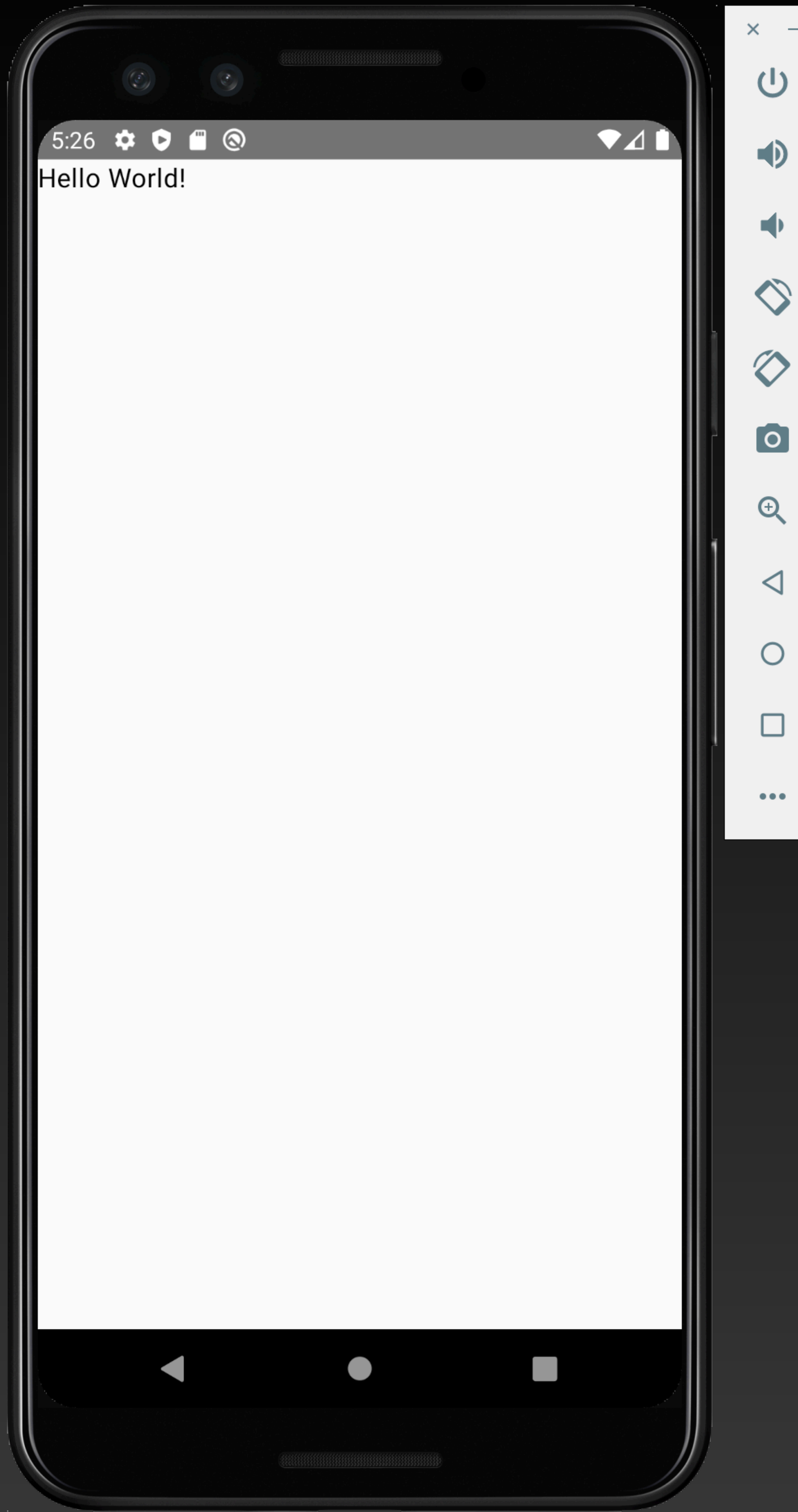
Events & State

```kotlin
class MainActivity: AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
      LandingScreen()
    }
  }

}
```

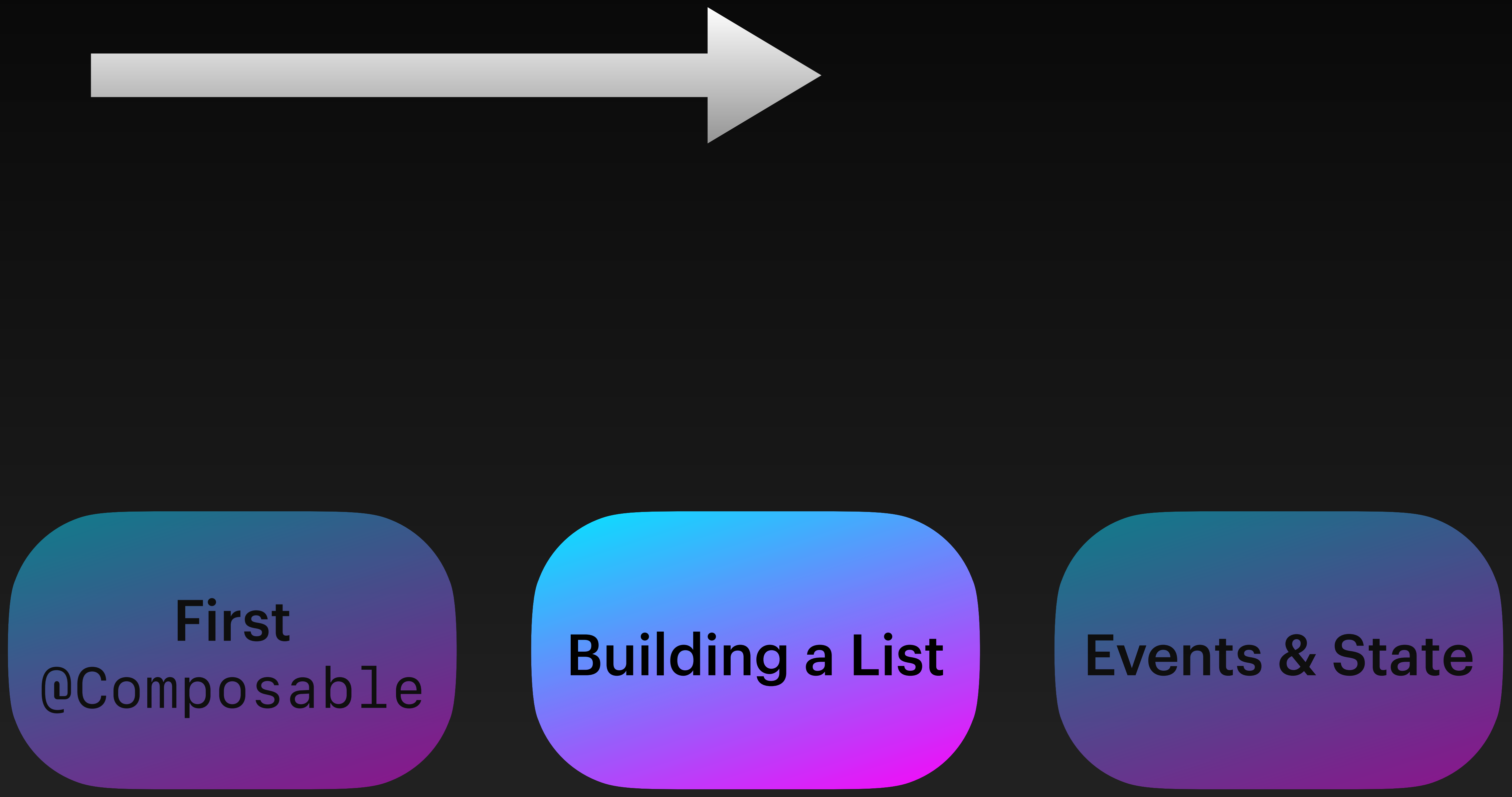
```
@Preview
@Composable
fun DefaultPreview() {
  LandingScreen()
}
```



```kotlin
package com.example.hncompose

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.compose.Composable
import androidx.ui.core.setContent
import androidx.ui.foundation.Text
import androidx.ui.material.MaterialTheme
import androidx.ui.tooling.preview.Preview

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MaterialTheme {
                LandingScreen()
            }
        }
    }
}

@Composable
fun LandingScreen() {
    Text(text = "Hello World!")
}

@Preview
@Composable
fun DefaultPreview() {
    MaterialTheme {
        LandingScreen()
    }
}
```



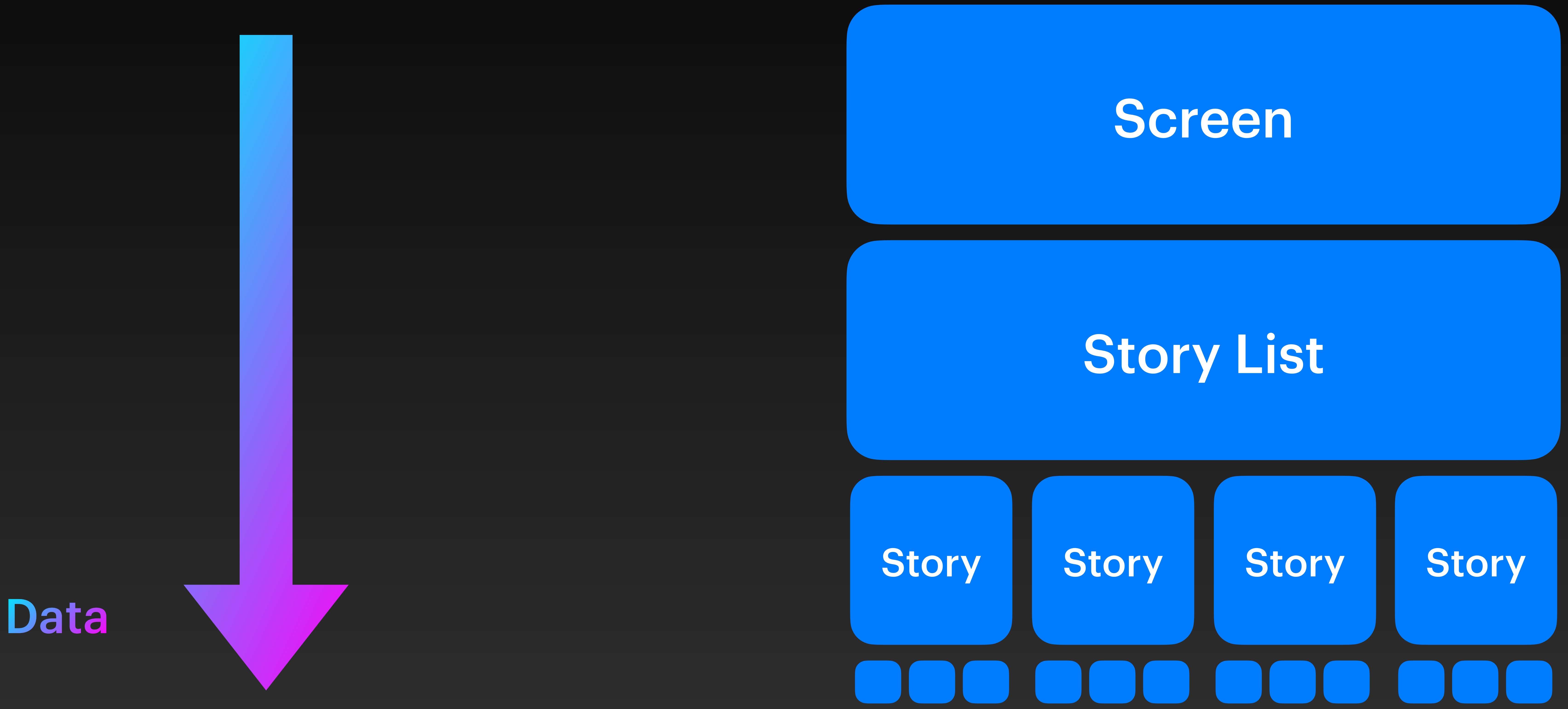
Flow



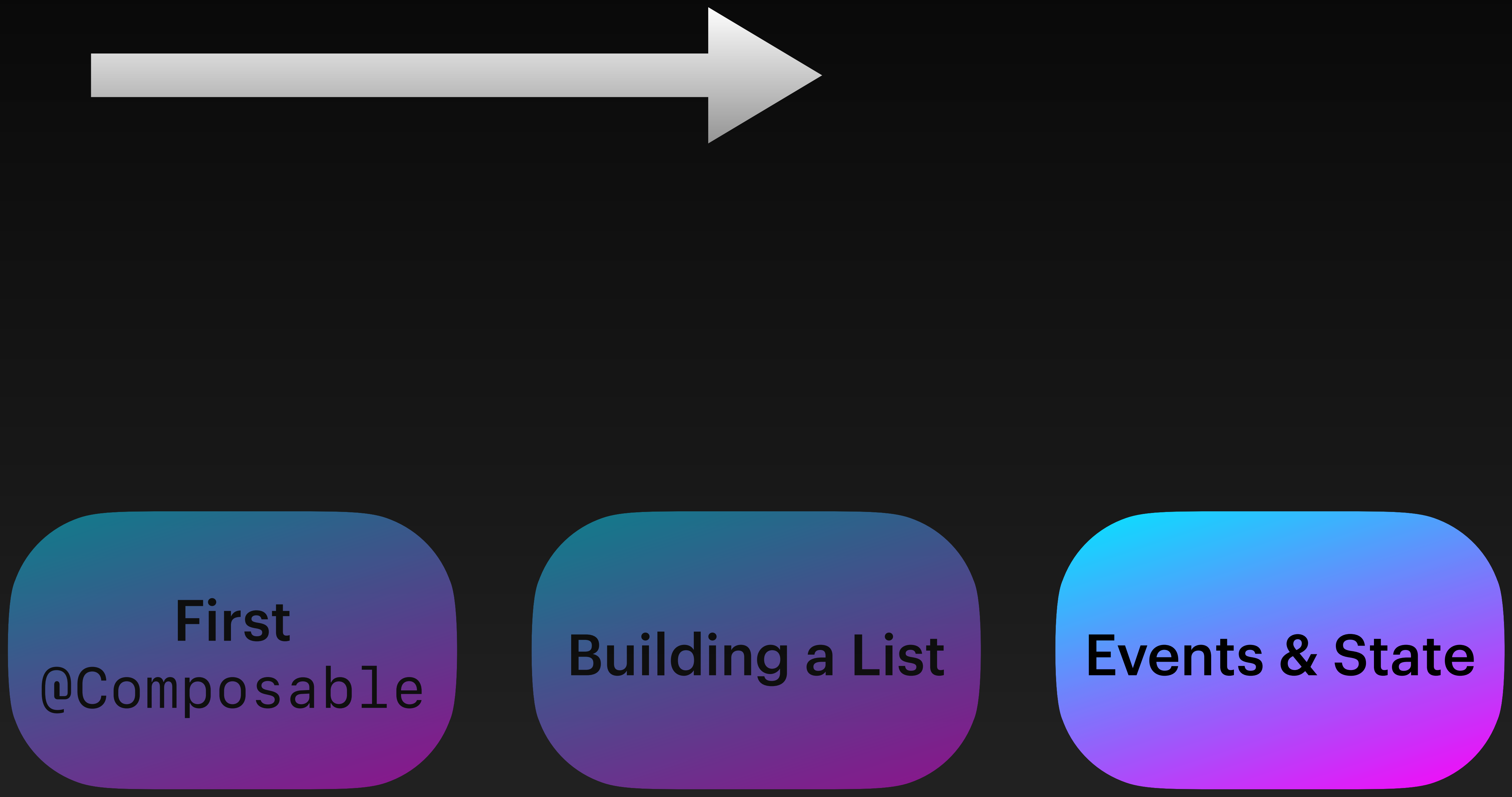
First `@Composable`

Building a List

Events & State


Data
Screen
Story List
Story
Story
Story
Story

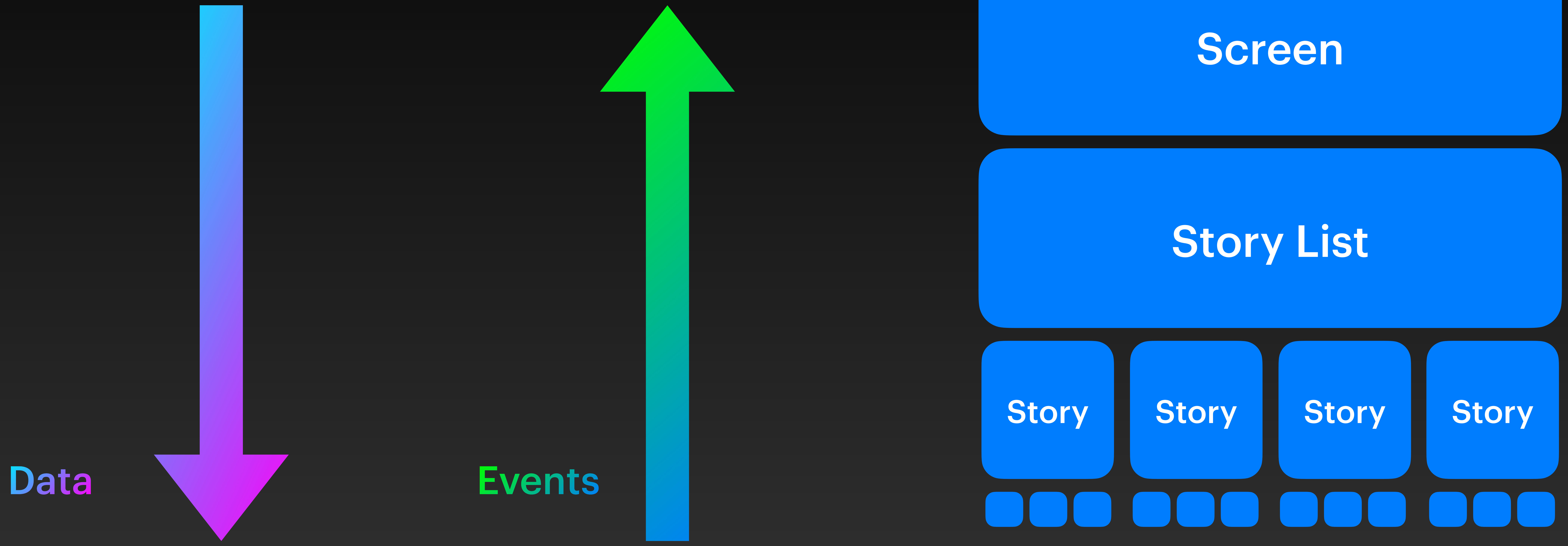
Flow



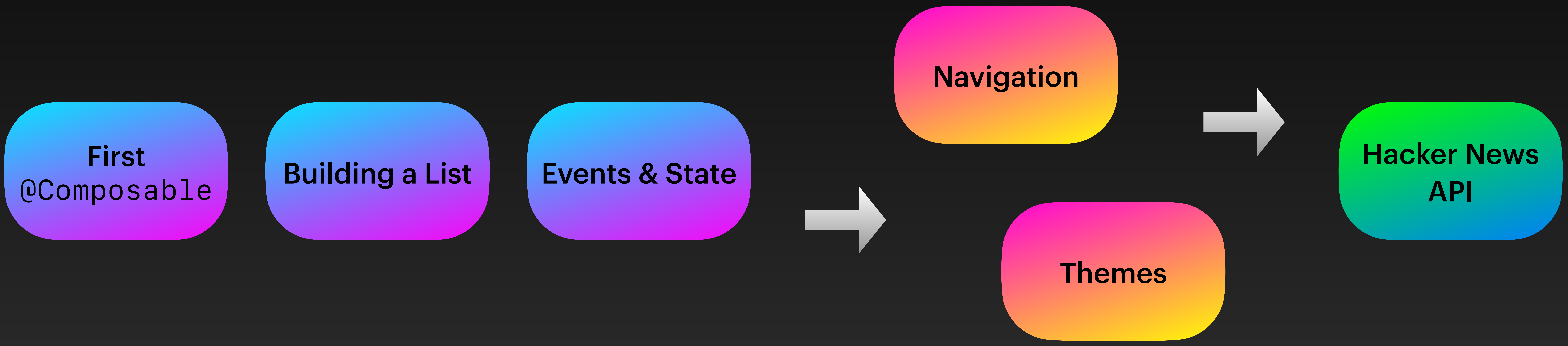
First @Composable

Building a List

Events & State


Data
Events
Screen
Story List
Story
Story
Story
Story

Flow



First
@Composable

Building a List

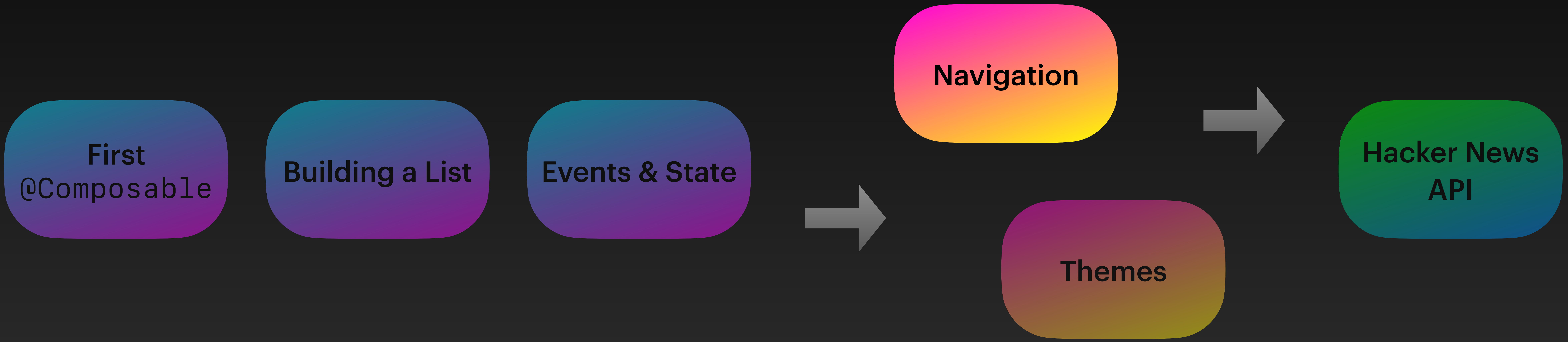
Events & State

Navigation

Themes

Hacker News
API

Flow



First `@Composable`

Building a List

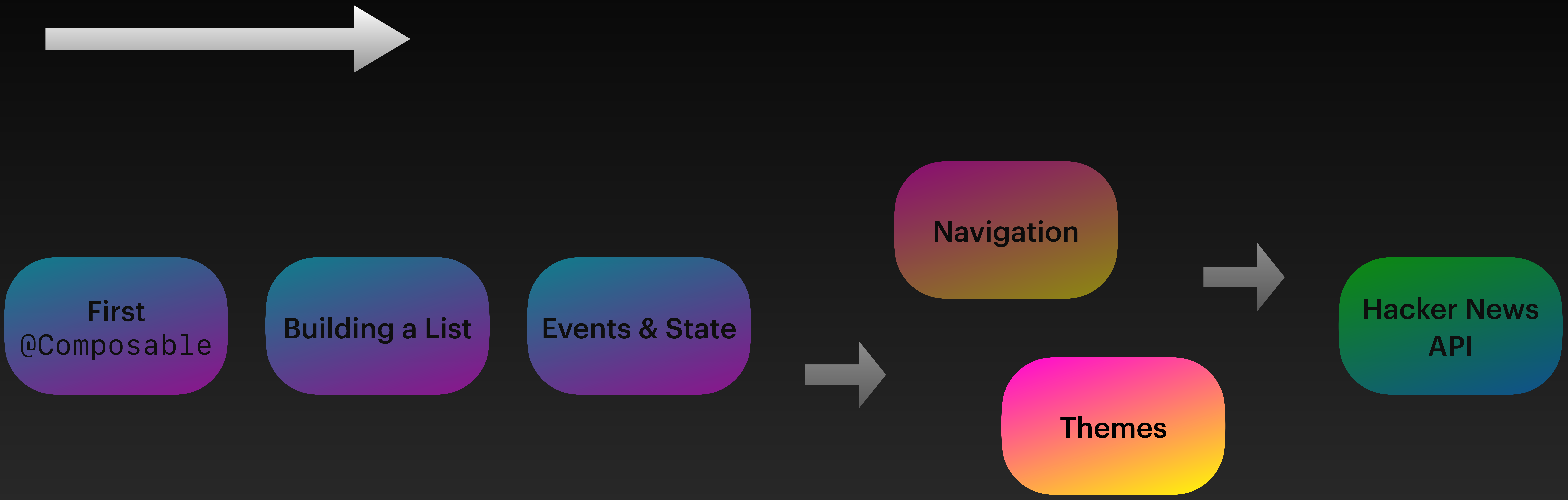
Events & State

Navigation

Themes

Hacker News API

Flow



First @Composable

Building a List

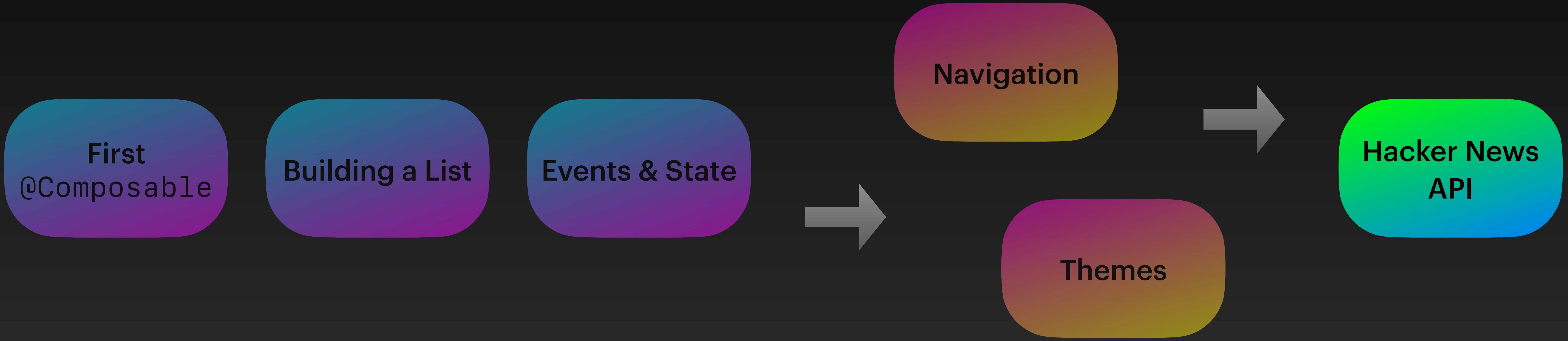
Events & State

Navigation

Themes

Hacker News API

Flow



First
`@Composable`

Building a List

Events & State

Navigation

Themes

Hacker News
API

Compose announced

May 2019

October 2019

Dev Preview 1.0

Dev Preview 2.0

~~June~~ July 2020

Summer 2020

Alpha

1.0

2021

Next Steps...

- To learn more about Compose
  - Google I/O 2018, 2019
  - ADS 2019
  - 11 Weeks of Android, 2020
- Google sample Compose app: JetNews (GitHub)
- Follow development with change logs
- Join the kotlinlang Slack team, and follow the #compose channel
- View links in the hn-compose repo

Thank You!