Jetpack Compose

План

1. Формат проведения **(мы тут)**
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose

Вопросы :)

План

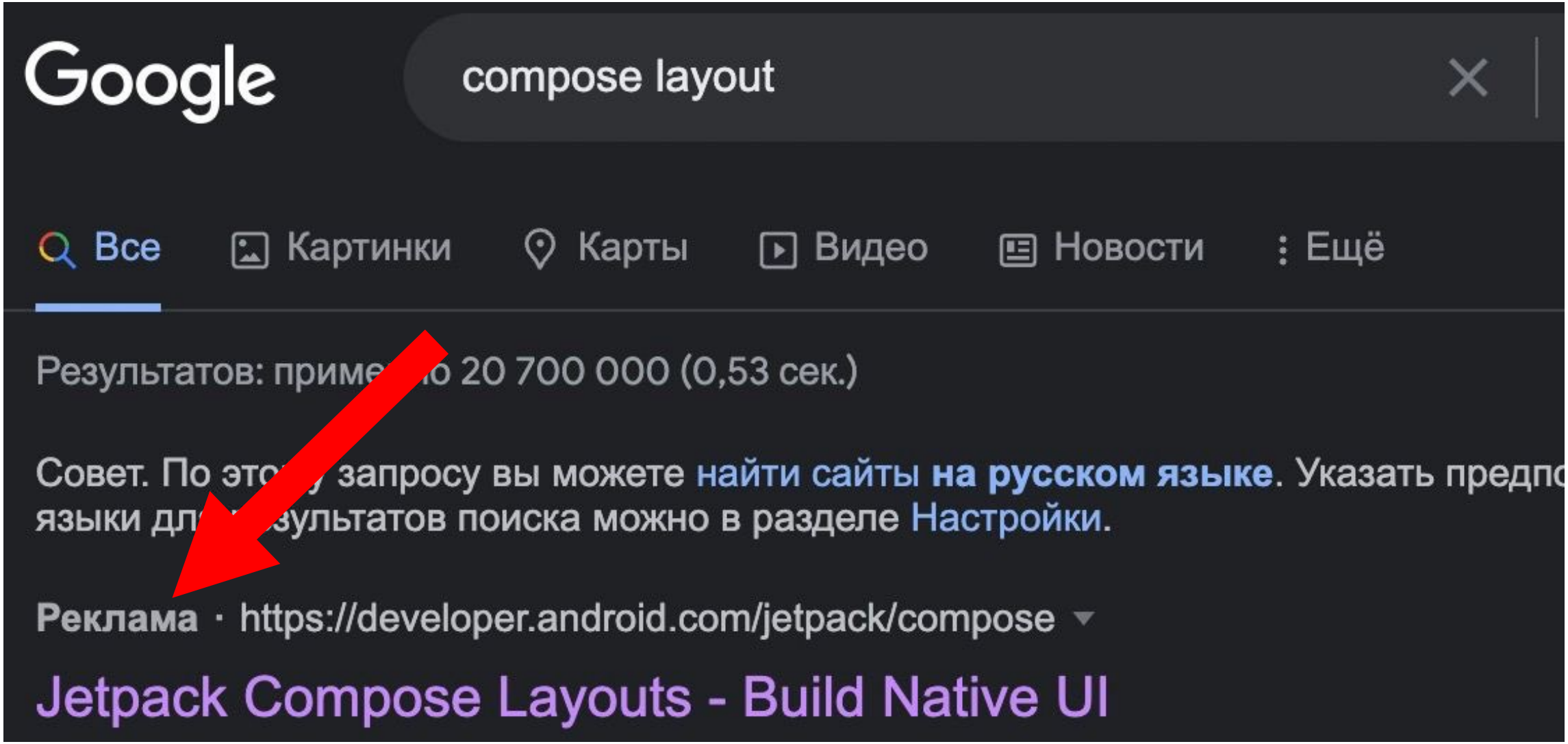
1. Формат проведения
2. Что такое Jetpack Compose **(мы тут)**
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose

Jetpack Compose is Android’s modern toolkit for building native UI

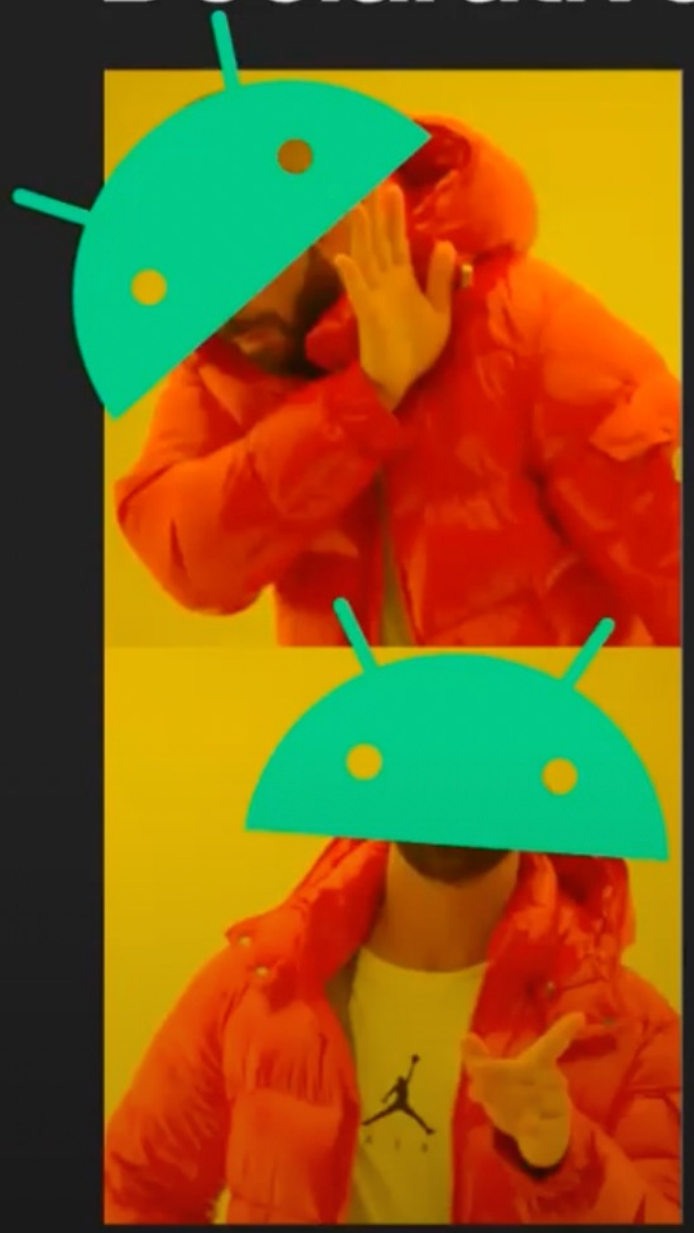
Jetpack Compose is Android's

modern

toolkit for building native UI


Google
compose layout
Все
Картинки
Карты
Видео
Новости
Ещё
Результатов: приме о 20 700 000 (0,53 сек.)
Совет. По это у запросу вы можете найти сайты на русском языке. Указать предпо
языки дл зультатов поиска можно в разделе Настройки.
Реклама · https://developer.android.com/jetpack/compose
Jetpack Compose Layouts - Build Native UI

Declarative UI



```
val imageView = ImageView(context, attrs, styleRes)
imageView.setImageResource(R.drawable.my_img)
val textView = TextView(context, attrs, styleRes)
textView.text = "Hello"
val linearLayout = LinearLayout(context, attrs)
linearLayout.addView(textView)
linearLayout.addView(imageView)
```

```
Column {
    Text(text = "Hello")
    Image(painter = painterResource(R.drawable.my_img))
}
```

Imperative UI

```
val imageView = ImageView(context, attrs, styleRes)
imageView.setImageResource(R.drawable.my_img)
val textView = TextView(context, attrs, styleRes)
textView.text = "Hello"
val linearLayout = LinearLayout(context, attrs)
linearLayout.addView(textView)
linearLayout.addView(imageView)
```

Imperative UI

```
val imageView = ImageView(context, attrs, styleRes)
imageView.setImageResource(R.drawable.my_img)
val textView = TextView(context, attrs, styleRes)
textView.text = "Hello"
val linearLayout = LinearLayout(context, attrs)
linearLayout.addView(textView)
linearLayout.addView(imageView)
```

Imperative UI

```
val imageView = ImageView(context, attrs, styleRes)
imageView.setImageResource(R.drawable.my_img)
val textView = TextView(context, attrs, styleRes)
textView.text = "Hello"
val linearLayout = LinearLayout(context, attrs)
linearLayout.addView(textView)
linearLayout.addView(imageView)
```

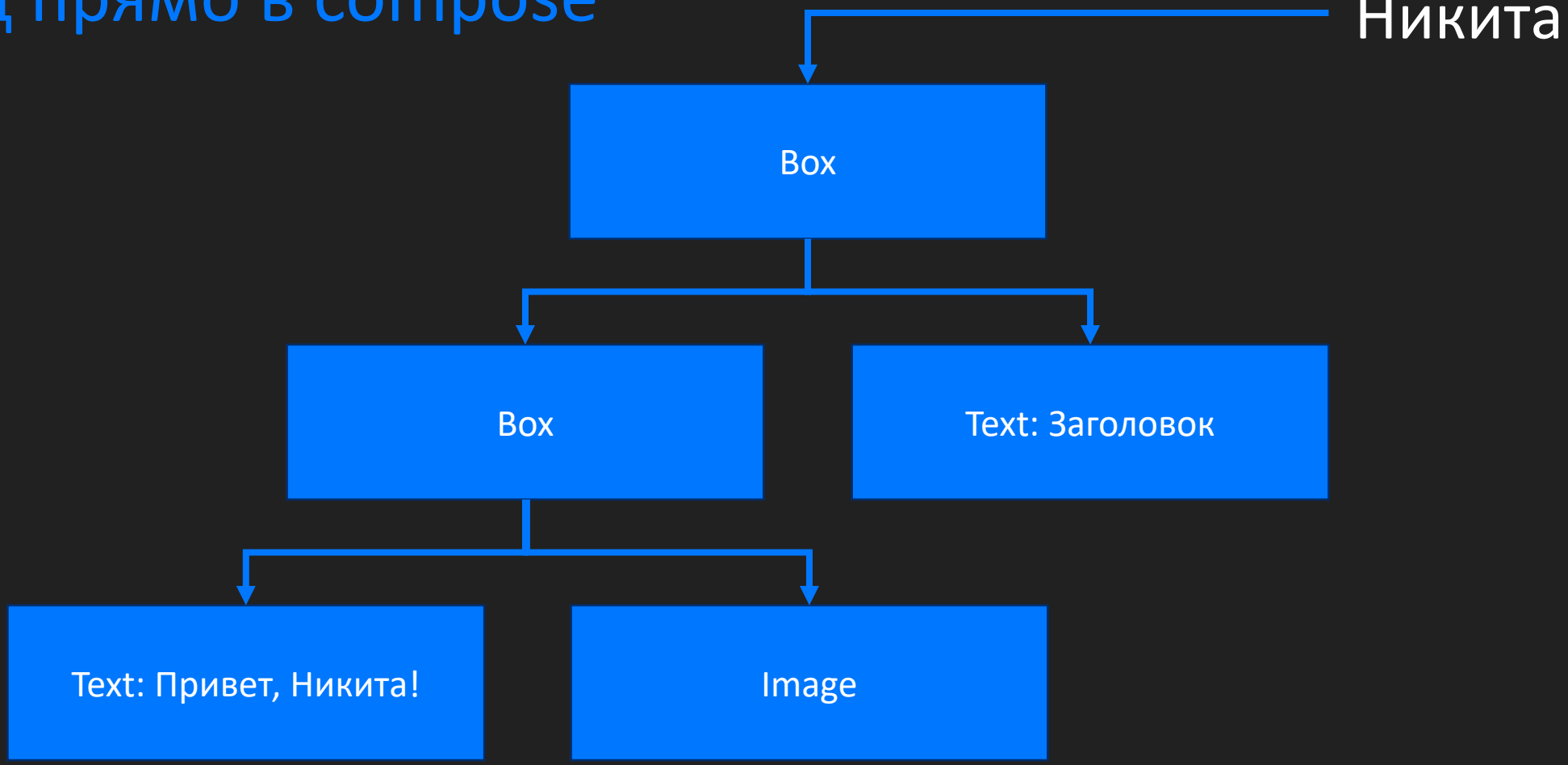
Declarative UI

```
Column {
    Text(text = "Hello")
    Image(painter = painterResource(R.drawable.my_img))
}
```

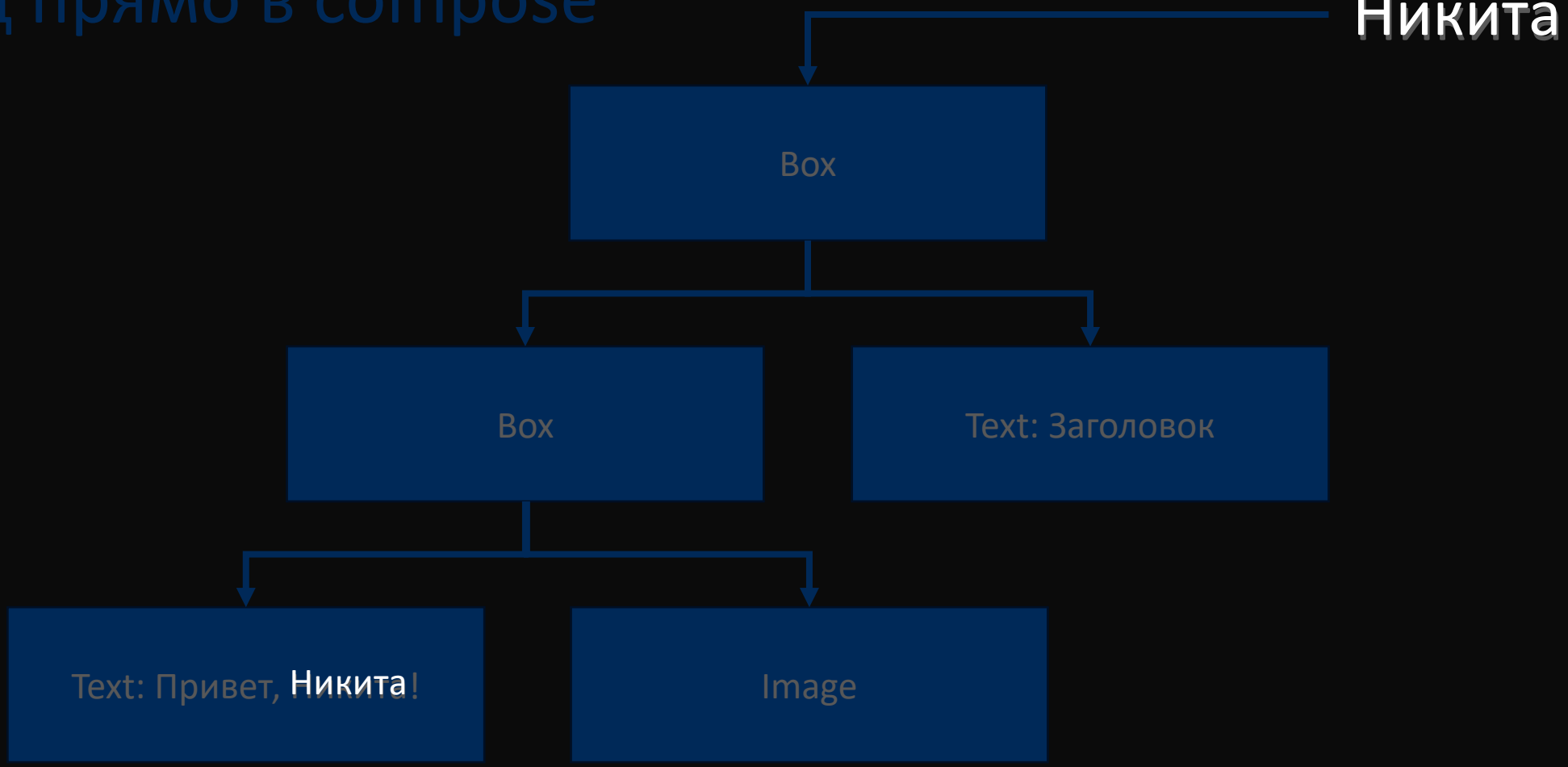
Код прямо в compose

```
@Composable
fun MyScreen(isActive: Boolean) {
    Box { this: BoxScope
        if (isActive) {
            Text( text: "Лампочка горит")
        } else {
            Text( text: "Лампочка не горит")
        }
    }
}
```

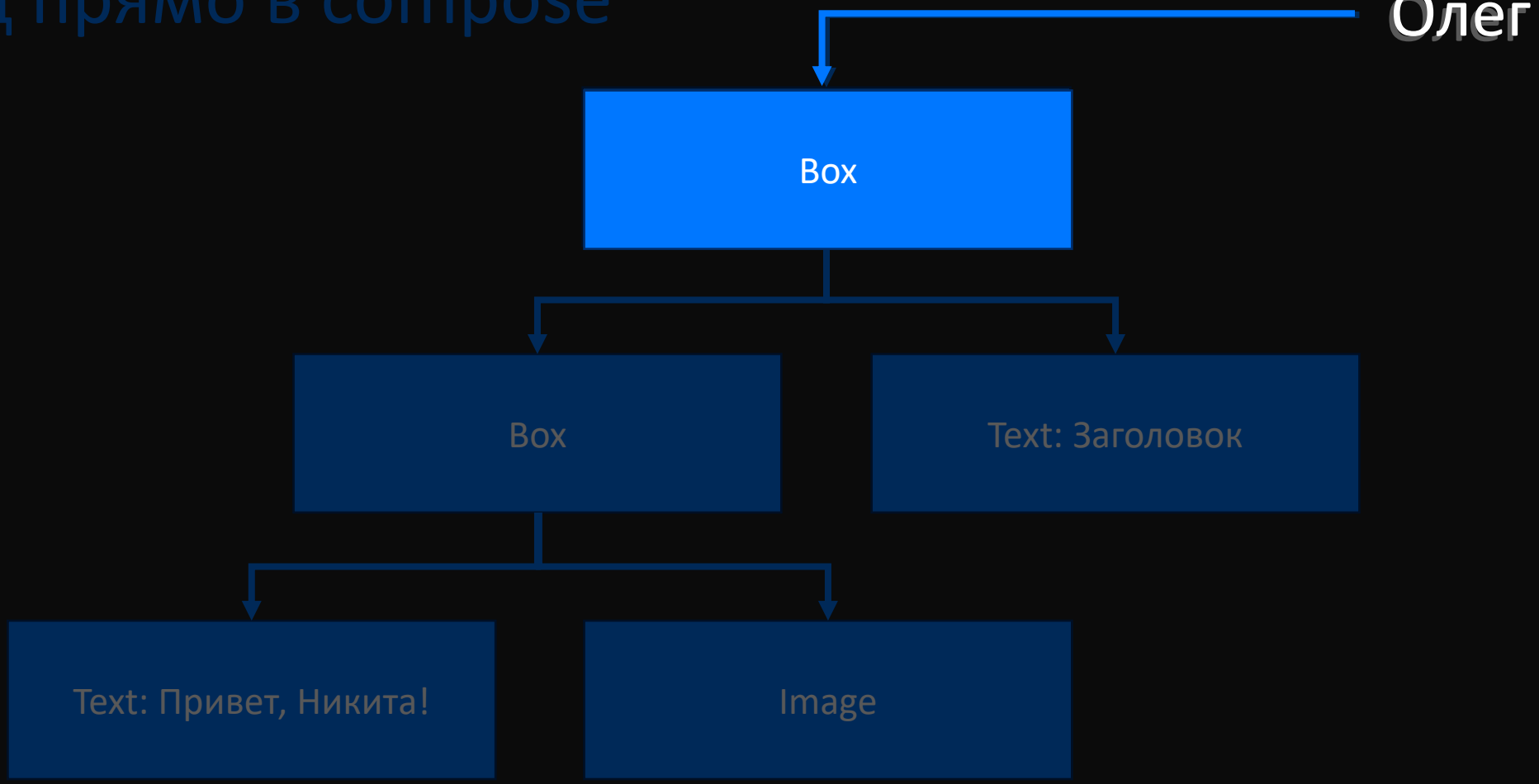
Код прямо в compose



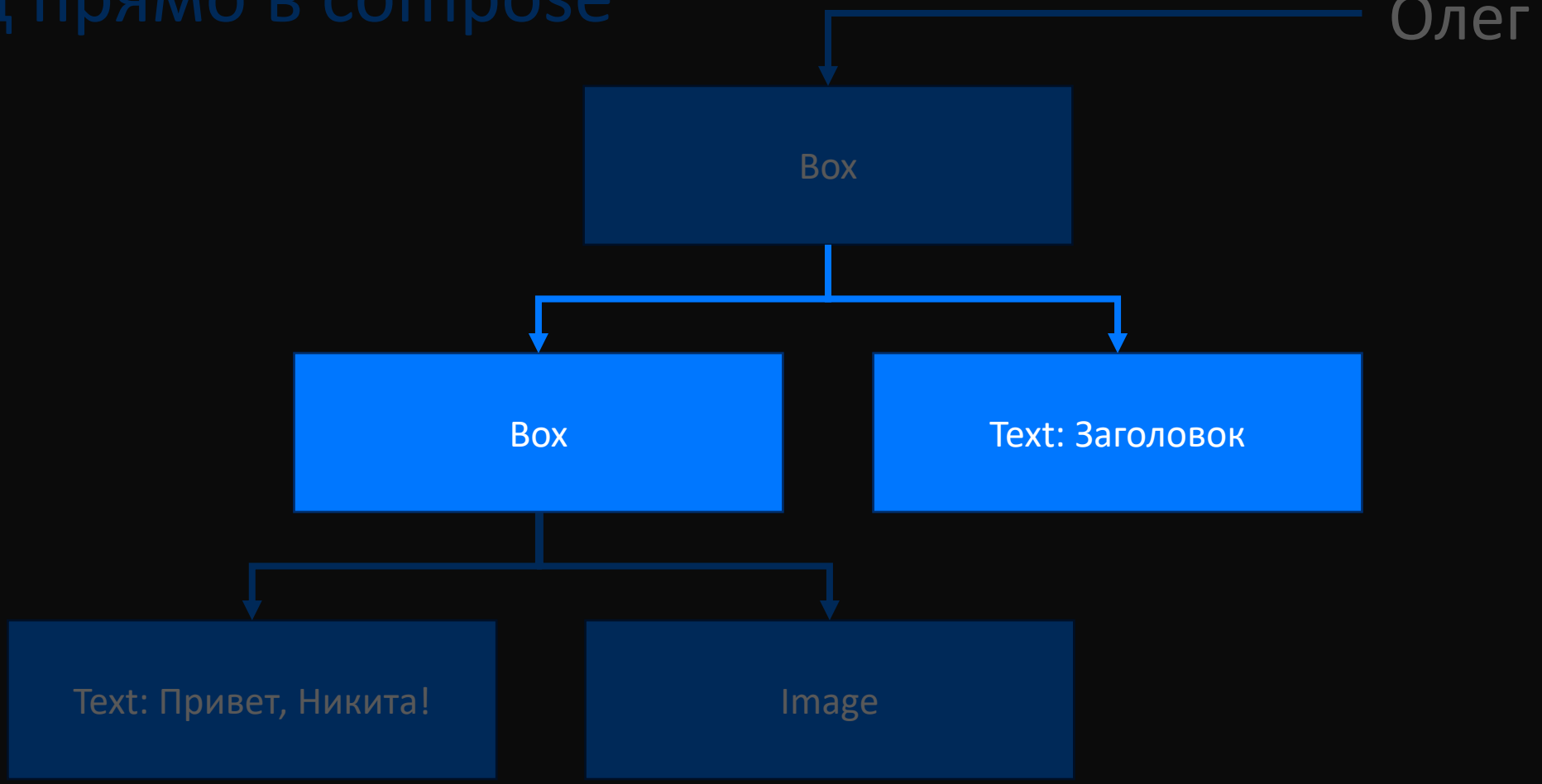
Код прямо в compose



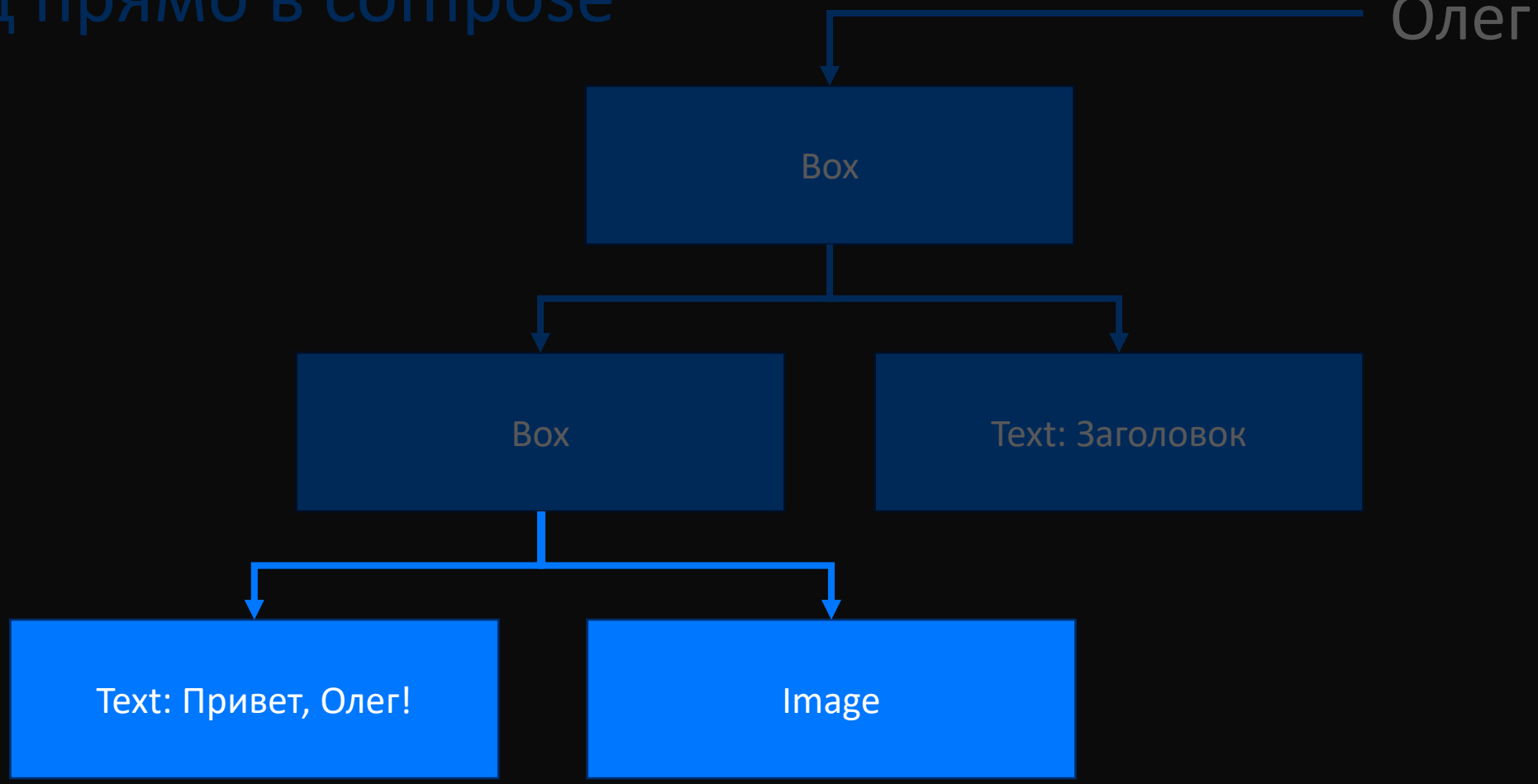
Код прямо в compose



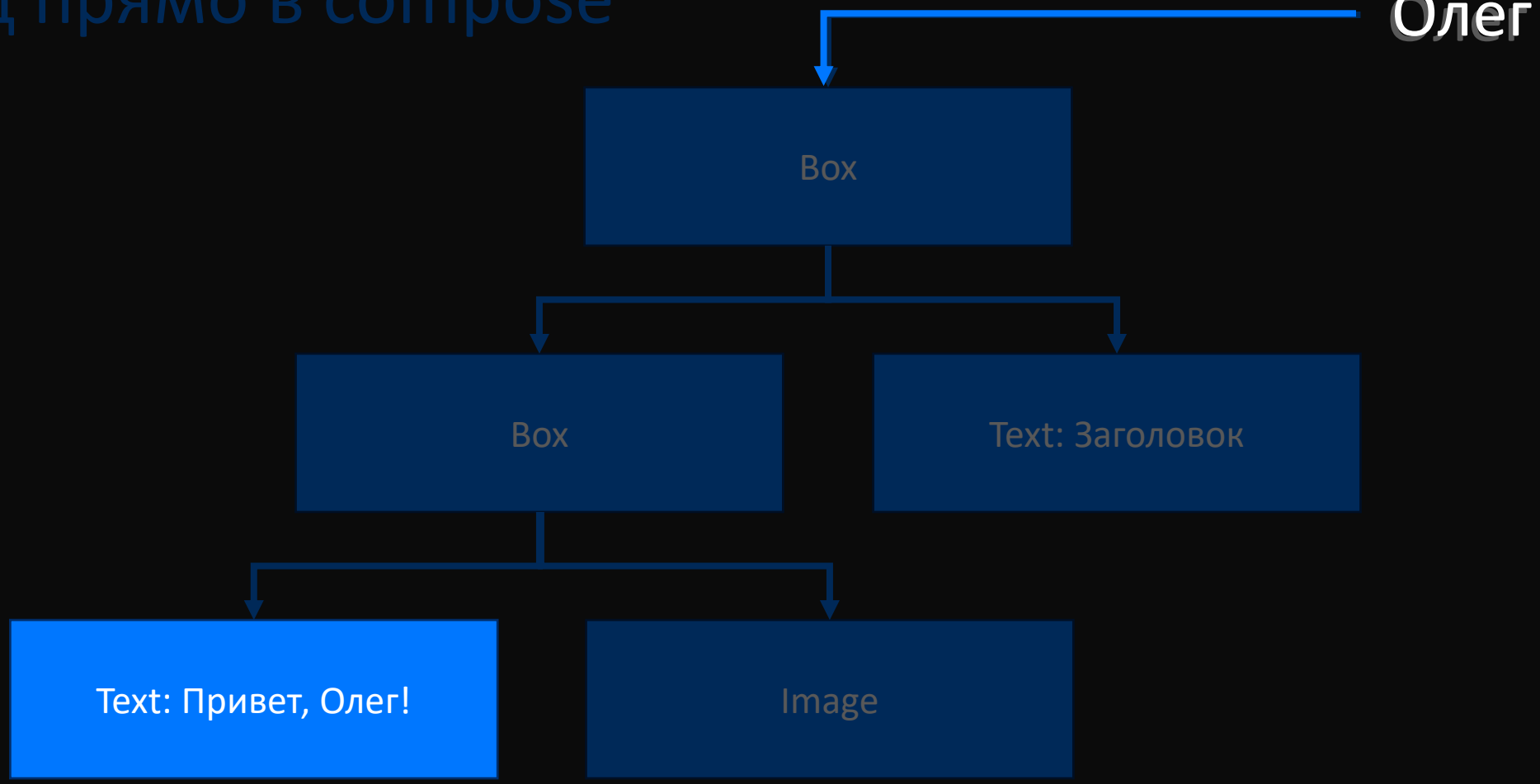
Код прямо в compose



Код прямо в compose



Код прямо в compose



Код прямо в compose

```
@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()
    }
    Updater<T>(currentComposer).update()
    SkippableUpdater<T>(currentComposer).skippableUpdate()
    currentComposer.startReplaceableGroup( key: 0x7ab4aae9)
    content()
    currentComposer.endReplaceableGroup()
    currentComposer.endNode()
}
```

Код прямо в compose

```
@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()
    }
    Updater<T>(currentComposer).update()
    SkippableUpdater<T>(currentComposer).skippableUpdate()
    currentComposer.startReplaceableGroup( key: 0x7ab4aae9)
    content()
    currentComposer.endReplaceableGroup()
    currentComposer.endNode()
}
```

Код прямо в compose

```
@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()
    }
    Updater<T>(currentComposer).update()
    SkippableUpdater<T>(currentComposer).skippableUpdate()
    currentComposer.startReplaceableGroup( key: 0x7ab4aae9)
    content()
    currentComposer.endReplaceableGroup()
    currentComposer.endNode()
}
```

Вопросы :)

План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я **(мы тут)**
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose

Куликов Никита

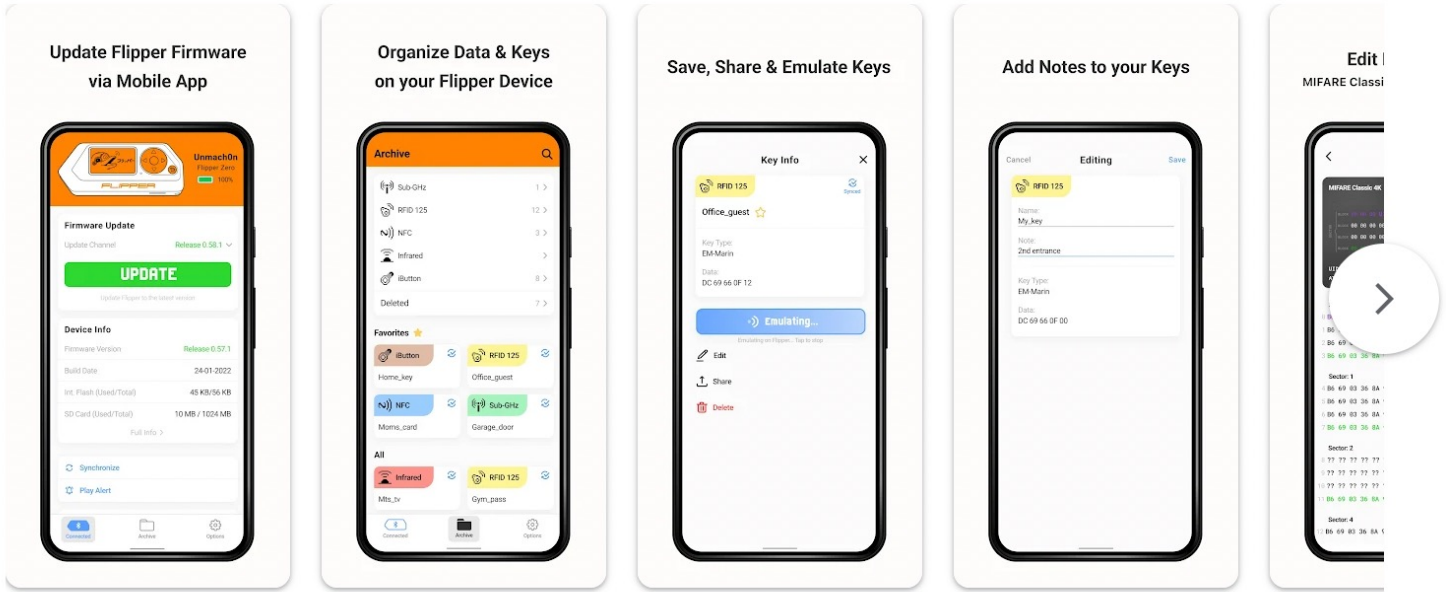
Head of Mobile in Flipper Devices

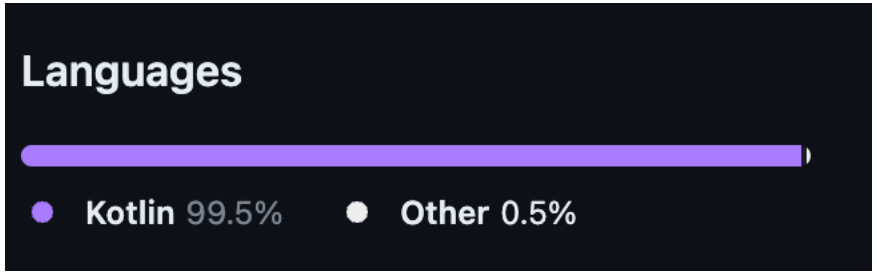
ex-Snapchat, ex-VK, ex-Yandex

UK Global Talent Visa Holder

Flipper Mobile App

Flipper Devices Inc.





https://github.com/Flipper-Zero/Flipper-Android-App

Вопросы :)

План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose **(мы тут)**
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose

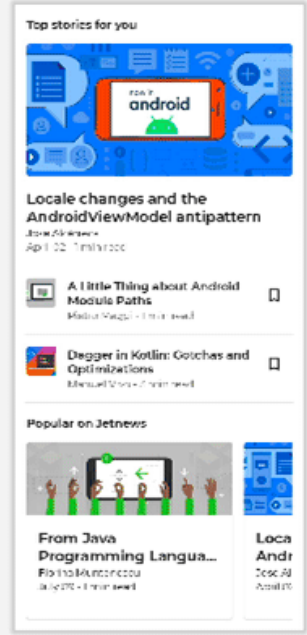
View Interop

```
@Composable
fun ViewInterop() {
    AndroidView(
        { context ->
            TextView(context)
        }
    )
}
```

Code Split Design

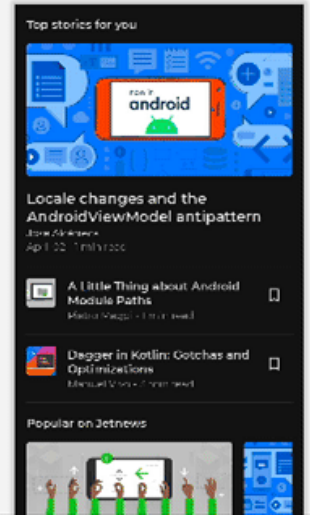
```
 */
@Composable
private fun PostListDivider() {
    Divider(
        modifier = Modifier.padding(horizontal = 14.dp),
        color = MaterialTheme.colors.onSurface.copy(alpha = 0.08f)
    )
}

@Preview( name: "Home screen body")
@Composable
fun PreviewHomeScreenBody() {
    ThemedPreview {
        val posts = loadFakePosts()
        PostList(posts, { }, setOf(), {})
    }
}

@Preview( name: "Home screen, open drawer")
@Composable
private fun PreviewDrawerOpen() {
    ThemedPreview {
        val scaffoldState = rememberScaffoldState(
            drawerState = rememberDrawerState(DrawerValue.Open)
        )
        HomeScreen(
            postsRepository = BlockingFakePostsRepository(LocalContext.current),
            scaffoldState = scaffoldState,
            navigateTo = { }
        )
    }
}

@Preview( name: "Home screen dark theme")
@Composable
fun PreviewHomeScreenBodyDark() {
    ThemedPreview(darkTheme = true) {
        val posts = loadFakePosts()
        PostList(posts, {}, setOf(), {})
    }
```

Home screen body

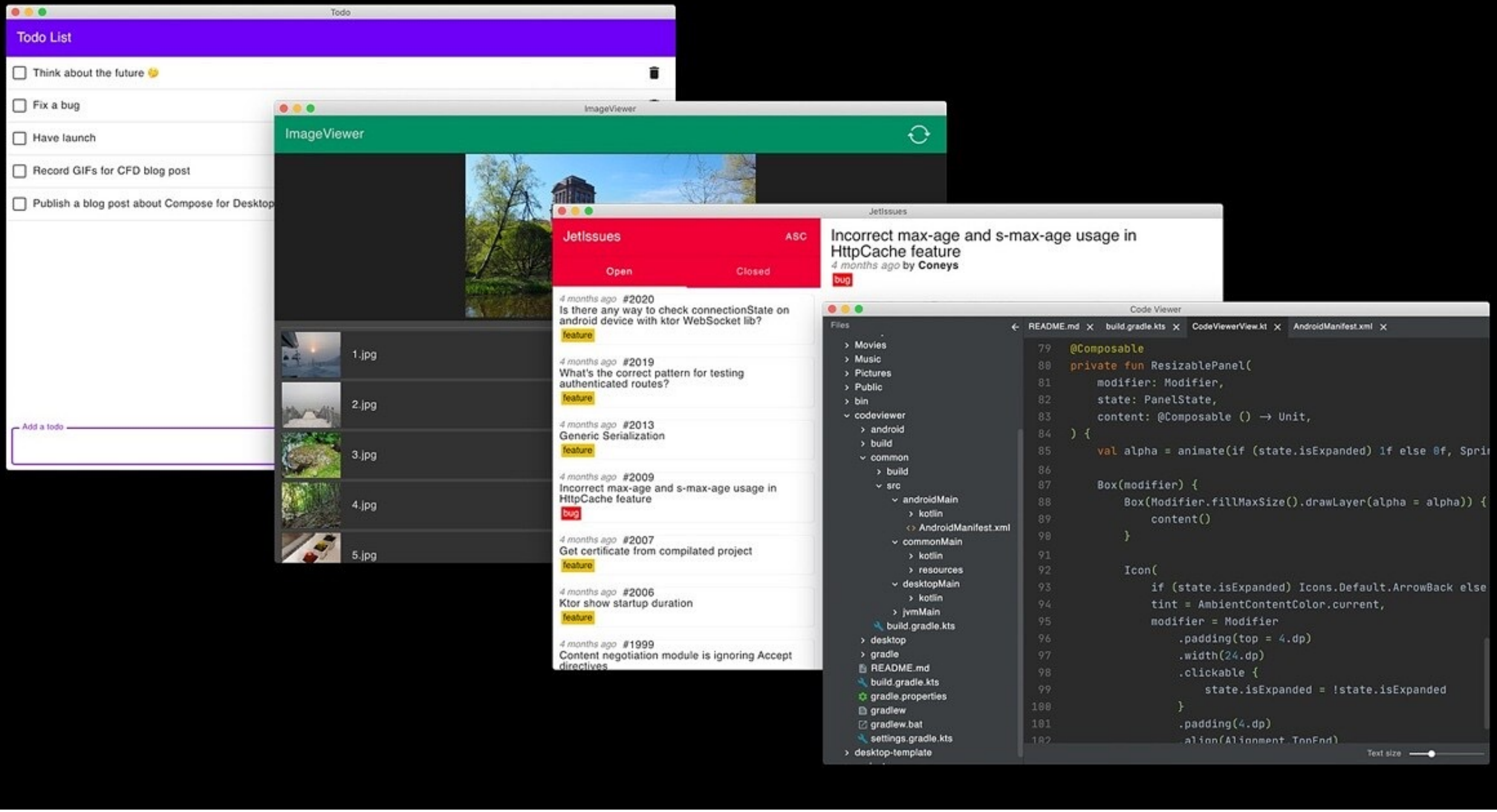


Home screen, ope...

Home screen dark ...



Home screen, open drawer dar...


Todo
Todo List
Think about the future
Fix a bug
Have launch
Record GIFs for CFD blog post
Publish a blog post about Compose for Desktop
Add a todo
ImageViewer
ImageViewer
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
JetIssues
JetIssues
ASC
Open
Closed
4 months ago #2020
Is there any way to check connectionState on android device with ktor WebSocket lib?
feature
4 months ago #2019
What's the correct pattern for testing authenticated routes?
feature
4 months ago #2013
Generic Serialization
feature
4 months ago #2009
Incorrect max-age and s-max-age usage in HttpCache feature
bug
4 months ago #2007
Get certificate from compilated project
feature
4 months ago #2006
Ktor show startup duration
feature
4 months ago #1999
Content negotiation module is ignoring Accept directives
Incorrect max-age and s-max-age usage in HttpCache feature
4 months ago by Coneys
bug
Code Viewer
Files
README.md
build.gradle.kts
CodeViewerView.kt
AndroidManifest.xml
Movies
Music
Pictures
Public
bin
codeviewer
android
build
common
build
src
androidMain
kotlin
AndroidManifest.xml
commonMain
kotlin
resources
desktopMain
kotlin
jvmMain
build.gradle.kts
desktop
gradle
README.md
build.gradle.kts
gradle.properties
gradlew
gradlew.bat
settings.gradle.kts
desktop-template
@Composable
private fun ResizablePanel(
modifier: Modifier,
state: PanelState,
content: @Composable () → Unit,
) {
val alpha = animate(if (state.isExpanded) 1f else 0f, Spri
Box(modifier) {
Box(Modifier.fillMaxSize().drawLayer(alpha = alpha)) {
content()
}
Icon(
if (state.isExpanded) Icons.Default.ArrowBack else
tint = AmbientContentColor.current,
modifier = Modifier
.padding(top = 4.dp)
.width(24.dp)
.clickable {
state.isExpanded = !state.isExpanded
}
.padding(4.dp)
.align(Alignment.TopEnd)
Text size

Код прямо в Compose

```
@Composable
fun MyScreen(isActive: Boolean) {
    Box { this: BoxScope
        if (isActive) {
            Text( text: "Лампочка горит")
        } else {
            Text( text: "Лампочка не горит")
        }
    }
}
```

Compose MVVM

```
@Composable
fun MVVMComposable() {
    val testViewModel = viewModel<TestViewModel>()
    val input by testViewModel.stateFlow.collectAsState()
    Text( text: "Hello, $input")
}
```

Вопросы :)

План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose **(мы тут)**
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose

Activity

```
class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, World"/>
</FrameLayout>
```


Compose

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <androidx.compose.ui.platform.ComposeView
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</FrameLayout>
```

Activity

```
class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val composeView = findViewById<ComposeView>(R.id.compose_content)
        composeView.setContent {
            Text( text: "Hello, World!")
        }
    }
}
```

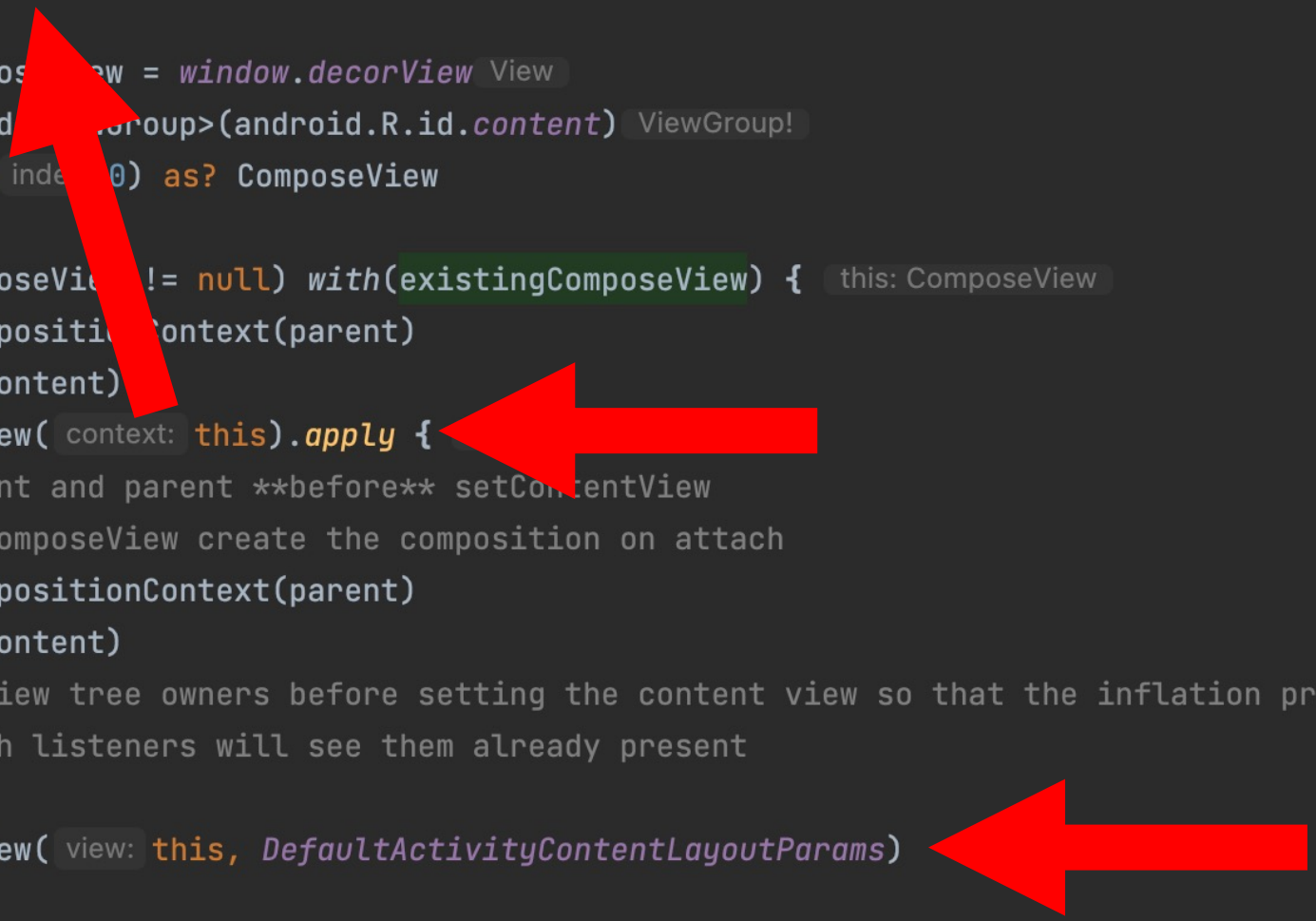
activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.compose.ui.platform.ComposeView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/compose_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

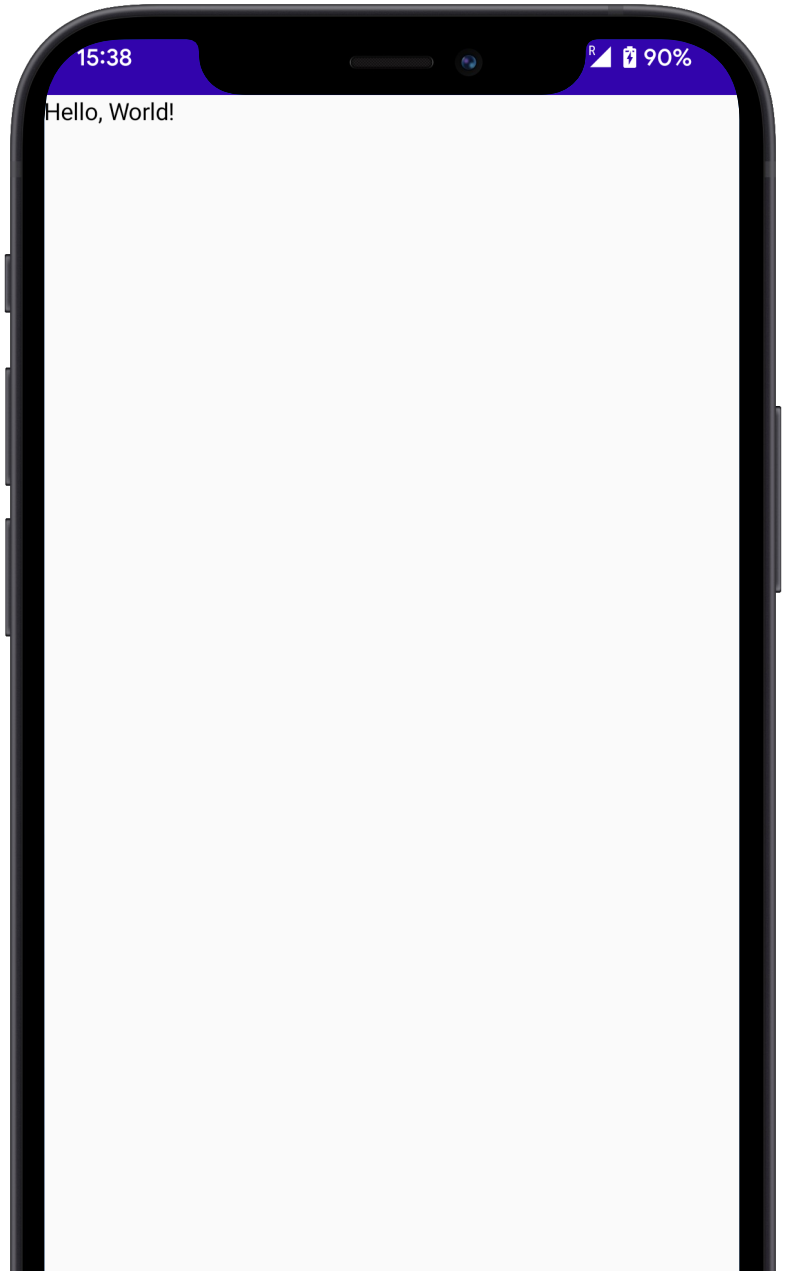
Activity

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text( text: "Hello, World!")
        }
    }
}
```

```
public fun ComponentActivity.setContent(
    parent: CompositionContext? = null,
    content: @Composable () -> Unit
) {
    val existingCompos  ew = window.decorView View
        .findViewById roup>(android.R.id.content) ViewGroup!
        .getChildAt( inde  0) as? ComposeView

    if (existingComposeVi  != null) with(existingComposeView) {  this: ComposeView
        setParentCompositi ontext(parent)
        setContent(content)
    } else ComposeView( context: this).apply {
        // Set content and parent **before** setCo entView
        // 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)
    }
}
```



Hello World!



Вопросы :)

План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose **(мы тут)**
7. Как работает Compose
8. MVVM на Jetpack Compose

Практика: tooling

Практика: containers

Modifier

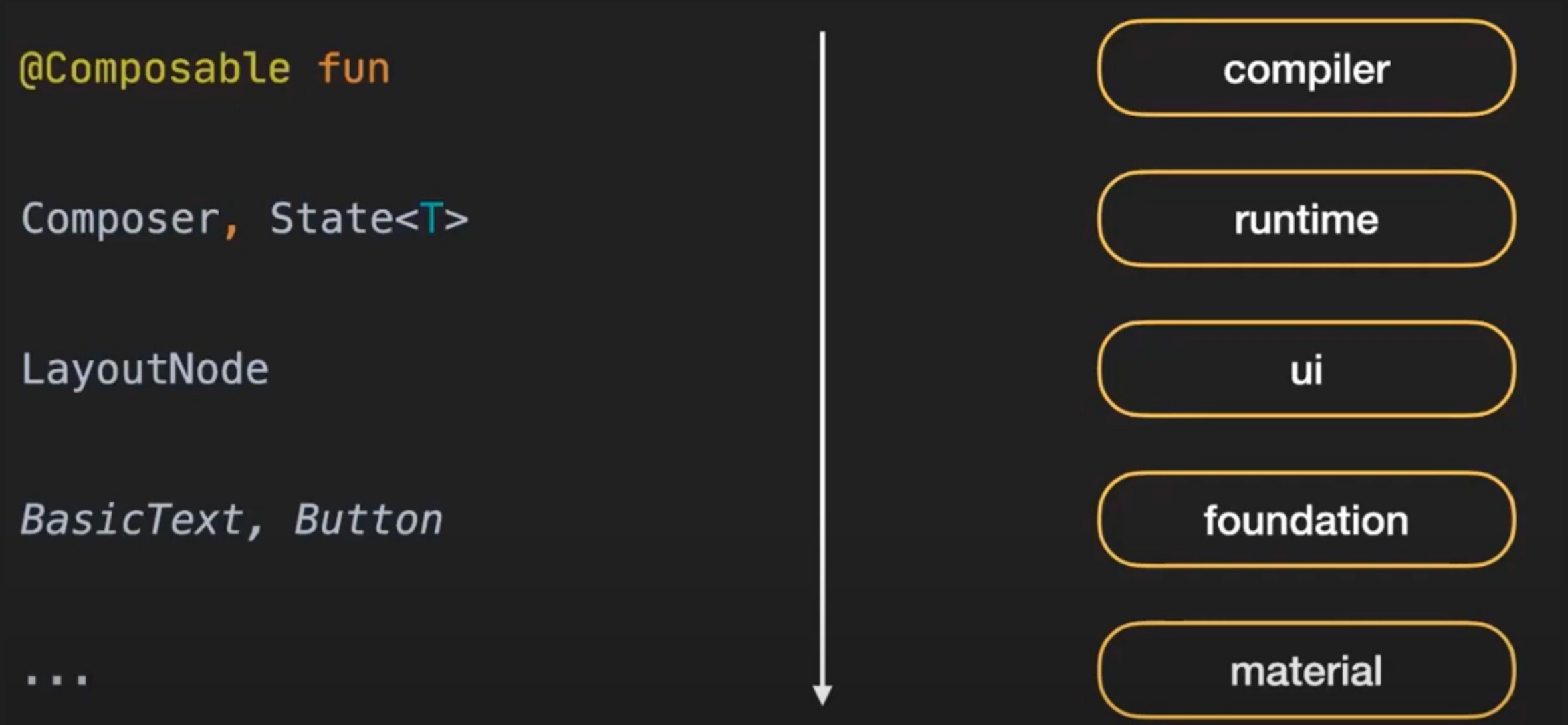
Modifiers allow you to decorate or augment a composable.
Modifiers let you do these sorts of things:

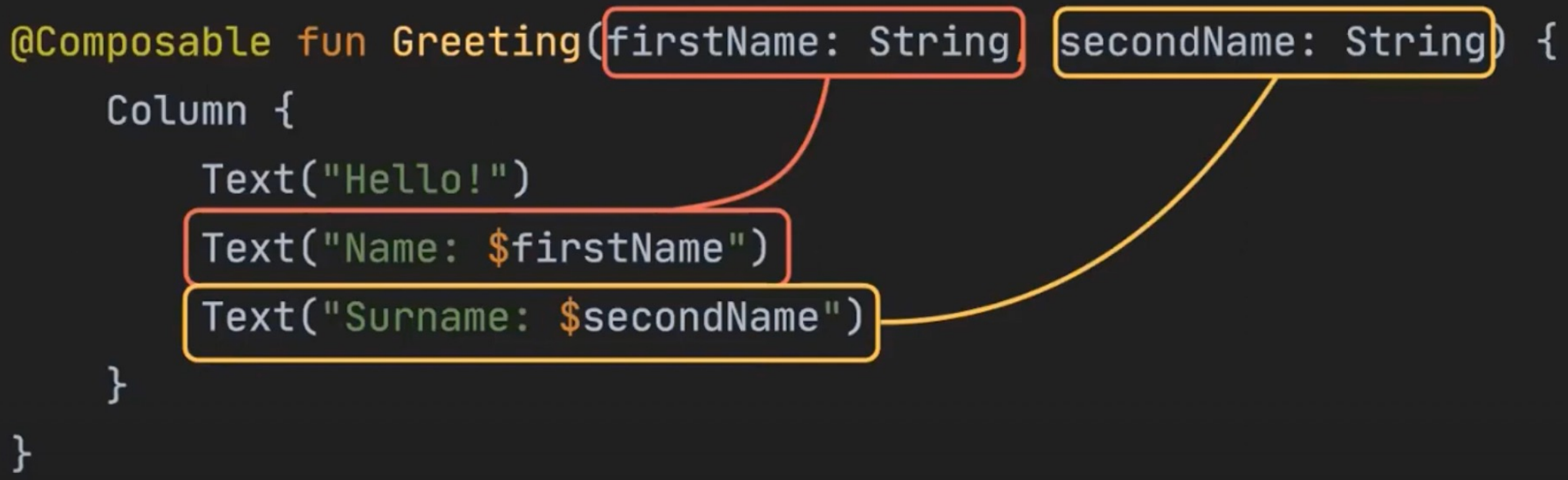
- Change the composable's **size**, layout, **behavior**, and **appearance**
- Add information, like accessibility labels
- Process **user input**
- Add high-level interactions, like making an element **clickable**, **scrollable**, **draggable**, or **zoomable**

Вопросы :)

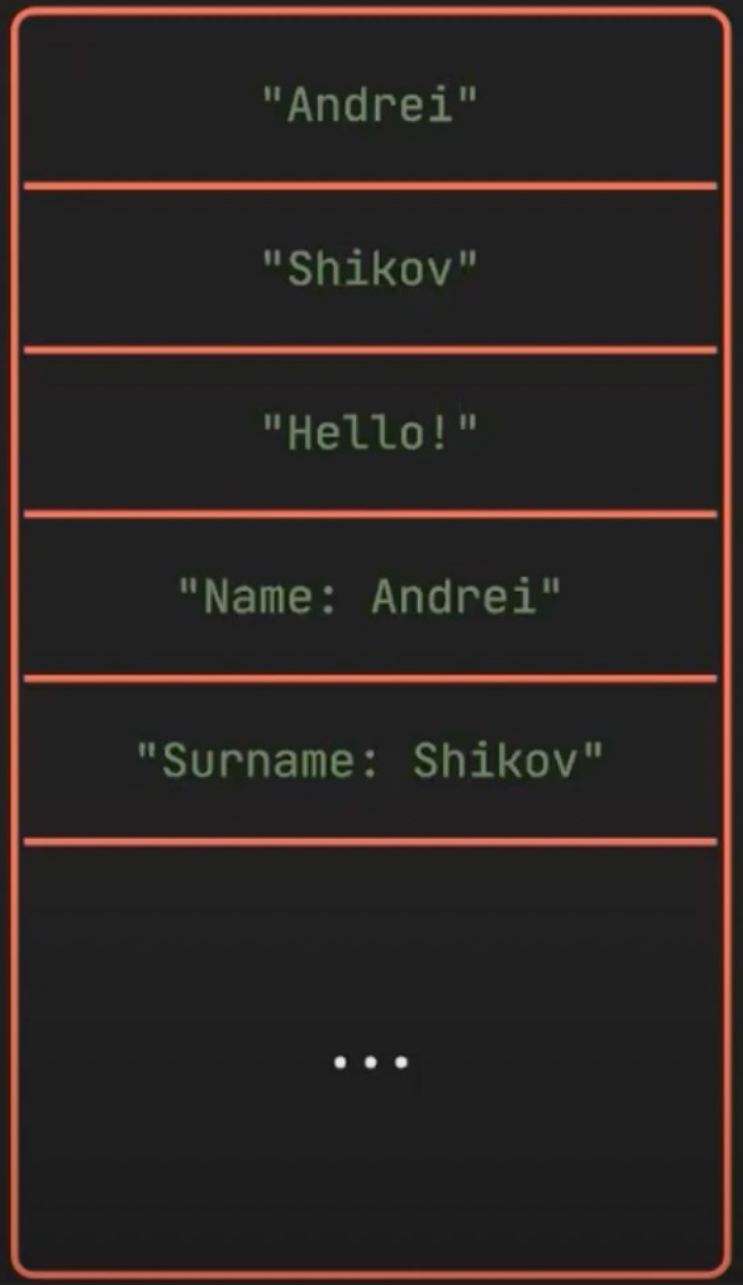
План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose **(мы тут)**
8. MVVM на Jetpack Compose


@Composable fun
Composer, State<T>
LayoutNode
BasicText, Button
...
compiler
runtime
ui
foundation
material


@Composable fun Greeting(firstName: String, secondName: String) {
Column {
Text("Hello!")
Text("Name: $firstName")
Text("Surname: $secondName")
}
}

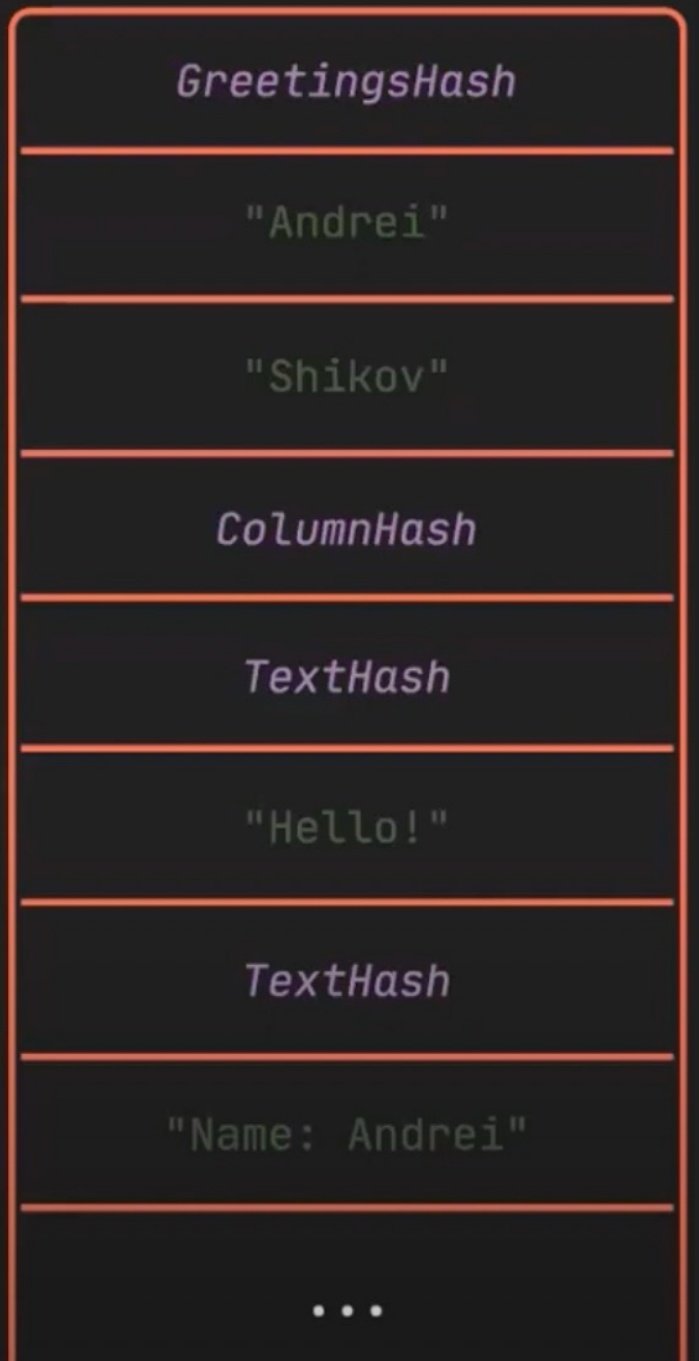
```
@Composable fun Greeting(
    firstName: String,
    secondName: String,
    $composer: Composer
) {
    val p1Changed = $composer.changed(firstName)
    val p2Changed = $composer.changed(secondName)
    Column {
        if (!$composer.skipping || p1Changed || p1Changed) {
            Text("Hello!", $composer)
            Text("Name: $firstName", $composer)
            Text("Surname: $secondName", $composer)
        }
    }
}
```



Вопросы :)

```
@Composable fun Greeting(
    firstName: String,
    secondName: String,
    $composer: Composer
) {
    $composer.startGroup(GreetingsHash)
    /* ... */
    Column {
        $composer.startGroup(ColumnHash)
        if (!$composer.skipping || p1Changed ...) {
            ...
        } else {
            composer.skipToGroupEnd()
        }
        $composer.endGroup()
    }
    $composer.endGroup()
}
```

```
@Composable fun Greeting(
    firstName: String,
    secondName: String,
    $composer: Composer
) {
    $composer.startGroup(GreetingsHash)
    /* ... */
    Column {
        $composer.startGroup(ColumnHash)
        if (!$composer.skipping || p1Changed ...) {
            ...
        } else {
            composer.skipToGroupEnd()
        }
        $composer.endGroup()
    }
    $composer.endGroup()
}
```

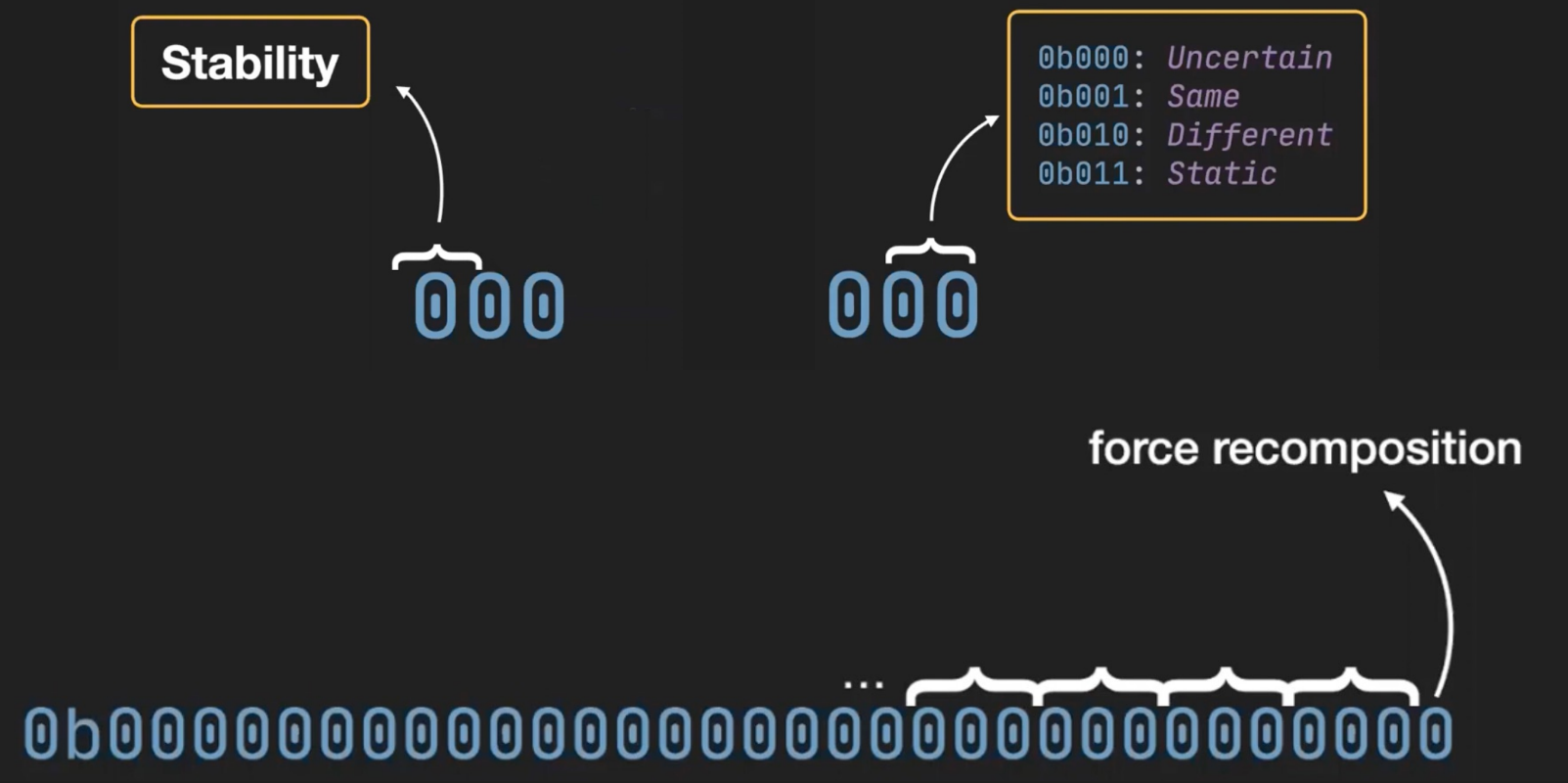


```
@Composable
fun Greeting(firstName: String, secondName: String, $composer: Composer?, $changed: Int) {
    $composer = $composer.startRestartGroup(-1339617867)
    sourceInformation($composer, "C(Greeting)7@291L134:Test.kt")
    val $dirty = $changed
    if ($changed and 0b1110 === 0) {
        $dirty = $dirty or if ($composer.changed(firstName)) 0b0100 else 0b0010
    }
    if ($changed and 0b01110000 === 0) {
        $dirty = $dirty or if ($composer.changed(secondName)) 0b00100000 else 0b00010000
    }
    if ($dirty and 0b01011011 !== 0b00010010 || !$composer.skipping) {
        Column({ $composer: Composer?, $changed: Int ->
            $composer.startReplaceableGroup(352554648)
            sourceInformation($composer, "C8@320L14,9@355L15,10@391L16:Test.kt")
            if ($changed and 0b1011 !== 0b0010 || !$composer.skipping) {
                Text("Hello!", $composer, 0b0110)
                Text("Name $firstName", $composer, 0b1110 and $dirty)
                Text("Surname: $secondName", $composer, 0b1110 and $dirty shr 3)
            } else {
                $composer.skipToGroupEnd()
            }
            $composer.endReplaceableGroup()
        }, $composer, 0)
    } else {
        $composer.skipToGroupEnd()
    }
    $composer.endRestartGroup()?.updateScope { $composer: Composer?, $force: Int ->
        Greeting(firstName, secondName, $composer, $changed or 0b0001)
    }
}
```

```
@Composable
fun Greeting(firstName: String, secondName: String, $composer: Composer?, $changed: Int) {
    $composer = $composer.startRestartGroup(-1339617867)
    sourceInformation($composer, "C(Greeting)7@291L134:Test.kt")
    val $dirty = $changed
    if ($changed and 0b1110 === 0) {
        $dirty = $dirty or if ($composer.changed(firstName)) 0b0100 else 0b0010
    }
    if ($changed and 0b01110000 === 0) {
        $dirty = $dirty or if ($composer.changed(secondName)) 0b00100000 else 0b00010000
    }
    if ($dirty and 0b01011011 !== 0b00010010 || !$composer.skipping) {
        Column({ $composer: Composer?, $changed: Int ->
            $composer.startReplaceableGroup(352554648)
            sourceInformation($composer, "C8@320L14,9@355L15,10@391L16:Test.kt")
            if ($changed and 0b1011 !== 0b0010 || !$composer.skipping) {
                Text("Hello!", $composer, 0b0110)
                Text("Name $firstName", $composer, 0b1110 and $dirty)
                Text("Surname: $secondName", $composer, 0b1110 and $dirty shr 3)
            } else {
                $composer.skipToGroupEnd()
            }
            $composer.endReplaceableGroup()
        }, $composer, 0)
    } else {
        $composer.skipToGroupEnd()
    }
    $composer.endRestartGroup()?.updateScope { $composer: Composer?, $force: Int ->
        Greeting(firstName, secondName, $composer, $changed or 0b0001)
    }
}
```

```
@Composable
fun Greeting(firstName: String, secondName: String, $composer: Composer?, $changed: Int) {
    $composer = $composer.startRestartGroup(-1339617867)
    sourceInformation($composer, "C(Greeting)7@291L134:Test.kt")
    val $dirty = $changed
    if ($changed and 0b1110 === 0) {
        $dirty = $dirty or if ($composer.changed(firstName)) 0b0100 else 0b0010
    }
    if ($changed and 0b01110000 === 0) {
        $dirty = $dirty or if ($composer.changed(secondName)) 0b00100000 else 0b00010000
    }
    if ($dirty and 0b01011011 !== 0b00010010 || !$composer.skipping) {
        Column({ $composer: Composer?, $changed: Int ->
            $composer.startReplaceableGroup(352554648)
            sourceInformation($composer, "C8@320L14,9@355L15,10@391L16:Test.kt")
            if ($changed and 0b1011 !== 0b0010 || !$composer.skipping) {
                Text("Hello!", $composer, 0b0110)
                Text("Name $firstName", $composer, 0b1110 and $dirty)
                Text("Surname: $secondName", $composer, 0b1110 and $dirty shr 3)
            } else {
                $composer.skipToGroupEnd()
            }
            $composer.endReplaceableGroup()
        }, $composer, 0)
    } else {
        $composer.skipToGroupEnd()
    }
    $composer.endRestartGroup()?.updateScope { $composer: Composer?, $force: Int ->
        Greeting(firstName, secondName, $composer, $changed or 0b0001)
    }
}
```

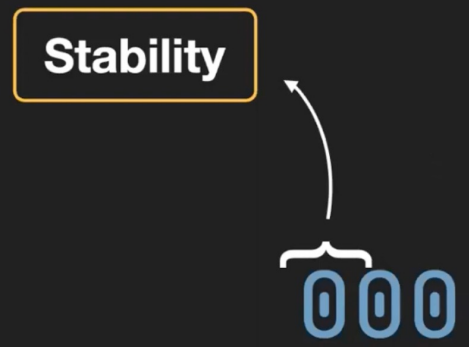
```
@Composable
fun Greeting(firstName: String, secondName: String, $composer: Composer?, $changed: Int) {
    $composer = $composer.startRestartGroup(-1339617867)
    sourceInformation($composer, "C(Greeting)7@291L134:Test.kt")
    val $dirty = $changed
    if ($changed and 0b1110 === 0) {
        $dirty = $dirty or if ($composer.changed(firstName)) 0b0100 else 0b0010
    }
    if ($changed and 0b01110000 === 0) {
        $dirty = $dirty or if ($composer.changed(secondName)) 0b00100000 else 0b00010000
    }
    if ($dirty and 0b01011011 !== 0b00010010 || !$composer.skipping) {
        Column({ $composer: Composer?, $changed: Int →
            $composer.startReplaceableGroup(352554648)
            sourceInformation($composer, "C8@320L14,9@355L15,10@391L16:Test.kt")
            if ($changed and 0b1011 !== 0b0010 || !$composer.skipping) {
                Text("Hello!", $composer, 0b0110)
                Text("Name $firstName", $composer, 0b1110 and $dirty)
                Text("Surname: $secondName", $composer, 0b1110 and $dirty shr 3)
            } else {
                $composer.skipToGroupEnd()
            }
            $composer.endReplaceableGroup()
        }, $composer, 0)
    } else {
        $composer.skipToGroupEnd()
    }
    $composer.endRestartGroup()?.updateScope { $composer: Composer?, $force: Int →
        Greeting(firstName, secondName, $composer, $changed or 0b0001)
    }
}
```


Stability
0b000: Uncertain
0b001: Same
0b010: Different
0b011: Static
000
000
force recomposition
...
0b00000000000000000000000000000000


"Andrei"
"Shikov"
"Hello!"
"Andrei"
"Shikov"
0b000: Uncertain
0b001: Same
0b010: Different
0b011: Static
000
"Andrei"
"Shikov"

```
data class MyClass(
    val items: List<String>,
    val isLoading: Boolean
)
```

```
data class MyClass(
    val items: MutableList<String>,
    var isLoading: Boolean
)
```



```
fun ComposableKeysList(onKeyOpen: (FlipperWearKey) -> Unit) {
    val keysListViewModel = viewModel<KeysListViewModel>()
    val state by keysListViewModel.getKeysListFlow().collectAsState()
    when (state) {
        is KeysListState.Loaded -> if (state.keys.isEmpty()) {
            
        } e
        Key
    }
}
```

Smart cast to 'KeysListState.Loaded' is impossible, because 'state' is a property that has open or cus

Cast expression 'state' to 'KeysListState.Loaded' ⌥⇧⏎ More actions... ⌥⏎

```
val state: KeysListState
```

```
@Stable
data class MyClass(
    val items: SnapshotStateList<String>,
    val isLoading: MutableState<Boolean>
)
```

Вопросы :)

План

1. Формат проведения
2. Что такое Jetpack Compose
3. Что такое я
4. Что может Jetpack Compose
5. Hello world на Jetpack Compose
6. Верстаем на Jetpack Compose
7. Как работает Compose
8. MVVM на Jetpack Compose **(мы тут)**

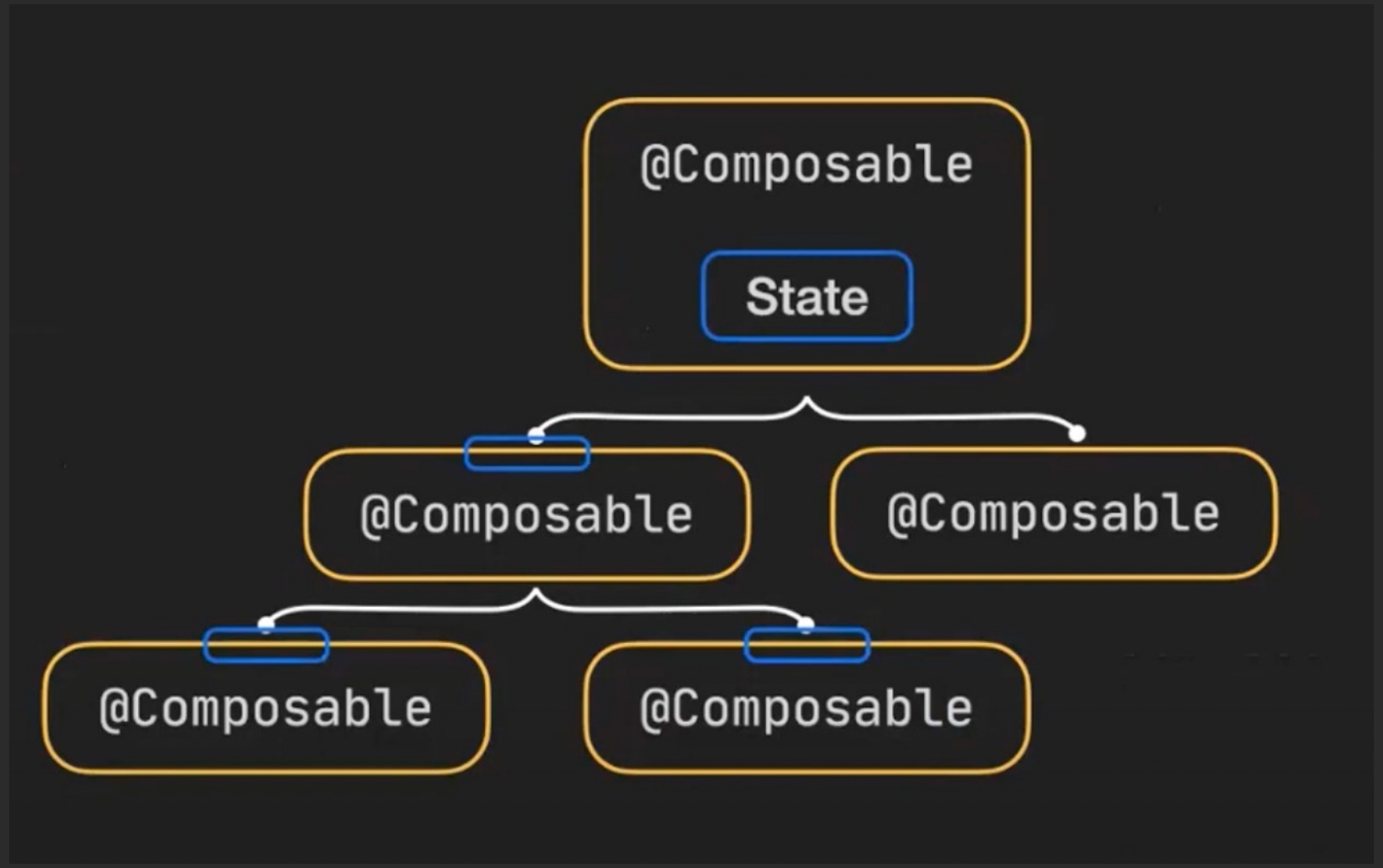
```
class CounterModel {
  val value = MutableStateFlow<Int>(0)

  fun increment() {
     value.update { count → count + 1 }
  }

  fun decrement() {
     value.update { count → count - 1 }
  }
}
```

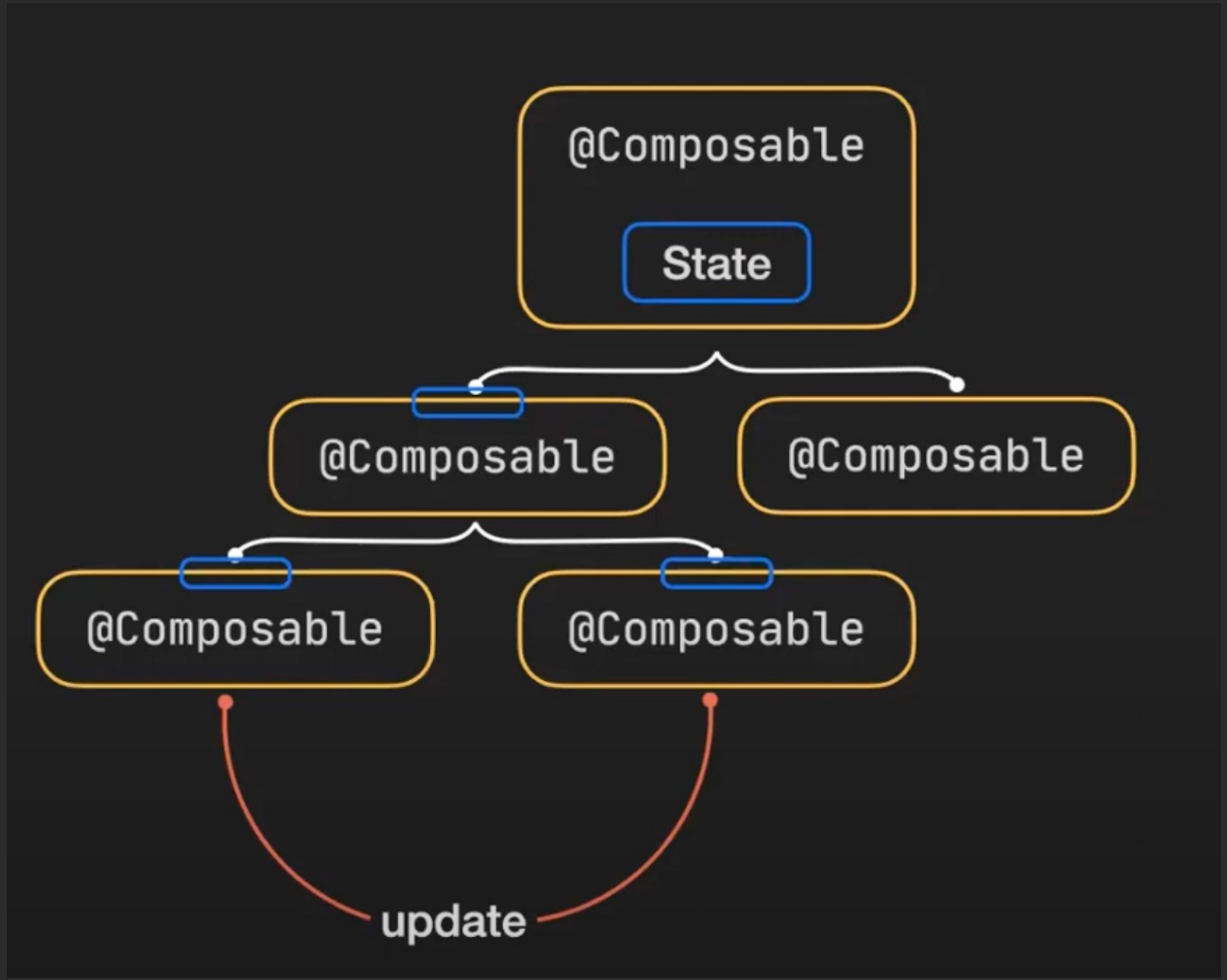
```
class CounterModel {
  var value by mutableStateOf<Int>(0)

  fun increment() {
      value += 1
  }

  fun decrement() {
      value -= 1
  }
}
```

```
class CounterModel {
  var value by mutableStateOf<Int>(0)
  var label by mutableStateOf<String>("Counter")
  /* ... */
}

class CounterView(private val model: CounterModel) {
  val textView = /* ... */

  init {
    snapshotFlow { refresh() }.launchIn(scope)
  }

  fun refresh() {
    textView.text = "${model.label}: ${model.value}"
  }
}
```

Практика: remember+state

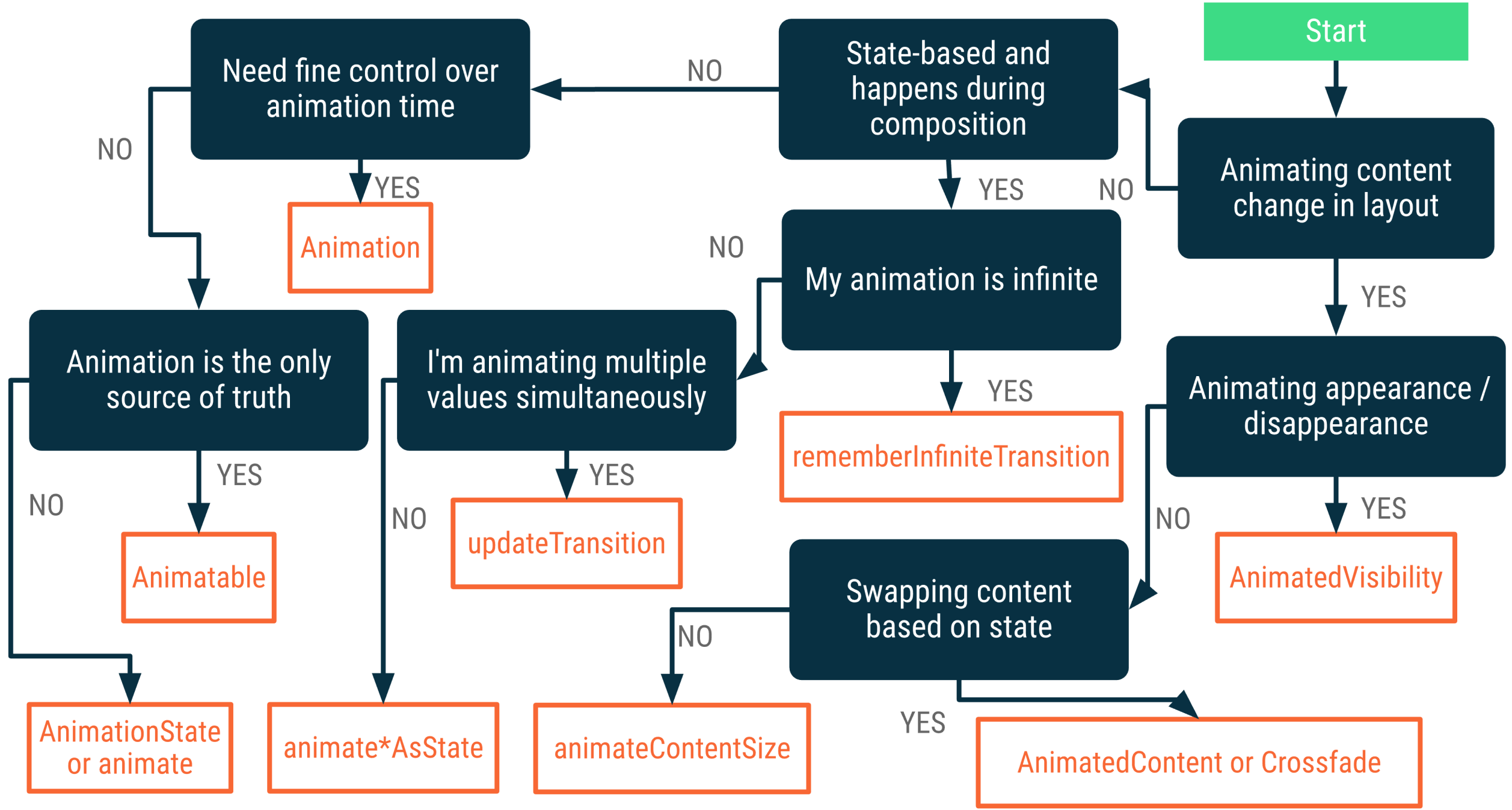
Вопросы :)


@Composable
State
@Composable
@Composable
@Composable
@Composable


@Composable
State
@Composable
@Composable
@Composable
@Composable
update

Вопросы :)

Практика+


Start
State-based and happens during composition
NO
Need fine control over animation time
YES
Animation
NO
Animation is the only source of truth
YES
Animatable
NO
AnimationState or animate
YES
My animation is infinite
NO
I'm animating multiple values simultaneously
YES
updateTransition
NO
animate*AsState
YES
rememberInfiniteTransition
NO
Swapping content based on state
YES
AnimatedContent or Crossfade
NO
animateContentSize
NO
Animating content change in layout
YES
Animating appearance / disappearance
YES
AnimatedVisibility
NO