Mash-Up 16th
2nd Seminar

2026.04.11

Fxxk Compose Early Return

```
@Composable
fun MyScreen() {
    return // ✕
```

Stop using “early return” in composable function

Android Team | 심은석

Index

01 문제 상황

Early Return

01 문제 상황

MyScreen.kt

```
@Composable
fun MyScreen(state: State stable ) {
    state.run {
        buildAnnotatedString {
            append(firstName ?: "")
            append(" ")
            append(lastName ?: "")
        }.also { text ->
            if (text.isBlank()) {
                return@run // 얼리 리턴!!
            }
        }.let {
            Text(text = "Hello $it!")
        }
    }
}
```

01 문제 상황

🤔 논리적으로는 맞아 보임

준비 안 됐으면 그냥 리턴하면 되지 않나?

💥 하지만 앱이 종료됨

ComposeRuntimeError와 함께 크래시 발생

😱 왜?

Compose 컴파일러 내부 동작 때문

02 에러 확인

⛔ RuntimeException Stack Trace

```
androidx.compose.runtime.ComposeRuntimeError: Compose Runtime internal error. Unexpected or incorrect
use of the Compose internal runtime API (Start/end imbalance).
at androidx.compose.runtime.ComposerKt.composeImmediateRuntimeError(Composer.kt:4822)
  at androidx.compose.runtime.ComposerImpl.finalizeCompose(Composer.kt:5350)
  at androidx.compose.runtime.ComposerImpl.endRoot(Composer.kt:1729)
  at androidx.compose.runtime.ComposerImpl.doCompose-aFTiNEg(Composer.kt:3858)
  at androidx.compose.runtime.ComposerImpl.composeContent--ZbOJvo$runtime(Composer.kt:3747)
  at androidx.compose.runtime.CompositionImpl.composeContent(Composition.kt:832)
  at androidx.compose.runtime.Recomposer.composeInitial$runtime(Recomposer.kt:1233)
…
```

02 에러 확인

요약

start와 end가 쌍을 이루지 못해
Compose runtime이 트리를 닫지 못하고 예외를 던진다.

03 컴파일러 동작 원리

컴파일러가 생성하는 코드 (의사코드)

```
composer.startGroup(key)


// ... 실제 컴포저블 로직 ...


composer.endGroup()
```

03 컴파일러 동작 원리

핵심 규칙

- 모든 startGroup → endGroup 쌍 필수
- ( ) 처럼 쌍을 이루어야 정상 → (()) OK
- ((닫히지 않으면 → RuntimeException

04 얼리 리턴이 문제인 이유

MyScreen.kt

```kotlin
@Composable
fun MyScreen(state: State stable ) {
    state.run {
        buildAnnotatedString {
            append(firstName ?: "")
            append(" ")
            append(lastName ?: "")
        }.also { text ->
            if (text.isBlank()) {
                return@run // 얼리 리턴!!
            }
        }.let {
            Text(text = "Hello $it!")
        }
    }
}
```

04 얼리 리턴이 문제인 이유

디컴파일된 코드 (Android Studio)

```
fun MyScreen(...) {

  $composer = $composer.startRestartGroup(1032080351); // ← START
  // ...
  $composer.startReplaceGroup(-308218668);
  // ...

  // text가 비어있지 않으면 실행
  if (!StringsKt.isBlank((CharSequence)text)) {
     // ...
     $composer.endReplaceGroup();
     // text 가 비어있으면 endRestartGroup 실행 X
  }
  // ...
  ScopeUpdateScope var10000 = $composer.endRestartGroup() // END

}
```

04 얼리 리턴이 문제인 이유

슬롯 테이블 상태

✅ 정상 흐름

```
startRestartGroup
  startReplaceGroup
  endReplaceGroup
endRestartGroup
```

(( )) ← 쌍 완성 ✅

❌ 얼리 리턴 흐름

```
startRestartGroup
  startReplaceGroup
  endReplaceGroup 없이 return!
endRestartGroup
```

(( ) ← 미완성 → 💥 Exception

04 얼리 리턴이 문제인 이유

결론

return이 endGroup을 건너뛰어 슬롯 테이블이 열린 채로 남음
→ RuntimeException

05 해결 방법

❌ Before – 얼리 리턴 사용

```
@Composable
fun MyScreen(state: State stable) {
    state.run {
        buildAnnotatedString {
            append(firstName ?: "")
            append(" ")
            append(lastName ?: "")
        }.also { text ->
            if (text.isBlank()) {
                return@run // 얼리 리턴!!
            }
        }.let {
            Text(text = "Hello $it!")
        }
    }
}
```

→

✅ After – return 제거

```
@Composable
fun MyScreen(state: State stable) {
    state.run {
        buildAnnotatedString {
            append(firstName ?: "")
            append(" ")
            append(lastName ?: "")
        }.also { text ->
            if (!text.isBlank()) {
                // 리턴 제거!!
                Text(text = "Hello $text!")
            }
        }
    }
}
```

정리하면

1. Compose 컴파일러는 composable 함수를 start/end 그룹 쌍으로 변환한다
2. 얼리 리턴은 endGroup을 건너뛰어 슬롯 테이블 구조를 깨뜨릴 수 있다
3. 결과: RuntimeException → 앱 크래시
4. 해결: return 제거, if 조건 반전, 별도 컴포저블 분리

얼리 리턴 대신, 조건을 뒤집자 ✅

06 Q&A

버전 이슈

Q1. 저는 얼리 리턴해도 괜찮던데 왜 그럴까요?

Compose compiler 버전을 확인해보세요.
이전 버전에서는 컴파일러가 얼리 리턴을 감지하면 자동으로 endGroup을 닫아줬어요.
최근 버전에서는 더 엄격하게 바뀌어 자동 처리 없이 그대로 런타임으로 넘깁니다.

또한 얼리 리턴을 했다고 무조건 크래시가 발생하는 것은 아닙니다.
함수 구조에 따라 start/end 그룹이 쌍을 이루는 형태가 유지된다면, 크래시는 발생하지 않습니다.
하지만 런타임 환경에 제어되기 때문에 가급적 사용하지 않는게 좋겠습니다. (본인의 책임)

Compose compiler changelog → developer.android.com/jetpack/compose/releases

06 Q&A

Q2. 왜 컴파일 타임이 아닌 런타임에서 터지나요?

런타임 에러

컴파일 단계에서는 이게 오류인지 판단하기 어렵기 때문이에요.
컴파일러는 코드를 정적으로만 분석하고, if 조건이 런타임에 실제로 true인지 알 수 없어요.
슬롯 테이블의 start/end 쌍 검증은 실행 중에 일어나기 때문에 런타임 예외로 발생합니다.

Compose compiler changelog → developer.android.com/jetpack/compose/releases

06 Q&A

```
@Composable
fun MyScreen(state: State  stable ) {
    state.run {
        buildAnnotatedString {
            append(firstName ?: "")
            append(" ")
            append(lastName ?: "")
        }.also { text ->
            if (text.isBlank()) {
                return@run // 얼리 리턴!!
            }
        }.let {
            Text(text = "Hello $it!")
        }
    }
}
```

만약 text가 채워져있다면

모든 그룹의 쌍이 맞기 때문에

Runtime 예외가 발생하지 않음

```
// text 가 비어있지 않아 endRestartGroup 실행
if (!StringsKt.isBlank((CharSequence)text)) {
      // ...
      $composer.endReplaceGroup();
  }
```

Compose compiler changelog → developer.android.com/jetpack/compose/releases

Thank you.

Android Team | 심은석