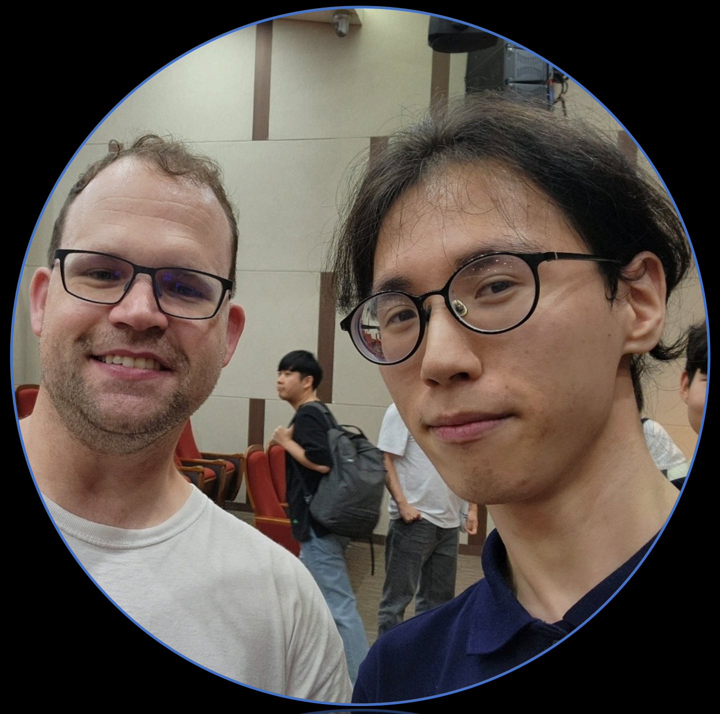
Kotlin User Groups Seoul

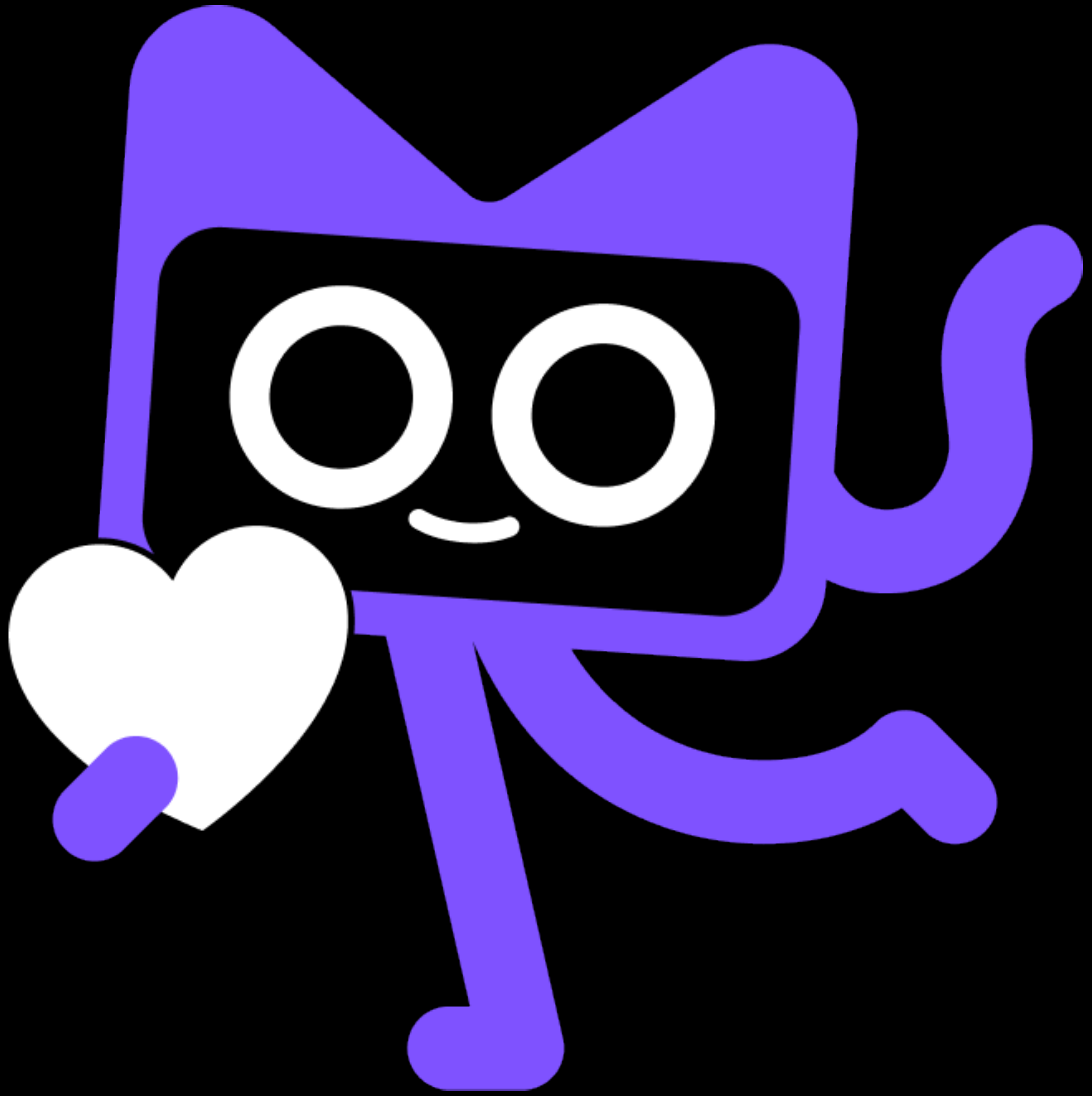
Kotlin Backend Meetup

Coroutine

코루틴 컴파일 과정 파헤치기



김찬우



섹션

1. Coroutine
2. Coroutine(Suspend) Lambda
3. State Machine
4. Coroutine Passing Style
5. Local Variables
6. Suspend Function

Coroutine

Coroutine

Kotlin 코루틴은 **비동기 프로그래밍**을 쉽게 구현할 수 있도록 도와주는 도구

Coroutine

Kotlin 코루틴은 **비동기 프로그래밍**을 쉽게 구현할 수 있도록 도와주는 도구

1. 경량 스레드
2. 직관적인 비동기 프로그래밍
3. 중단 가능한 함수

```kotlin
fun doSomething(): Unit
```

```kotlin
suspend fun doSomething(): Unit
```

Coroutine(Suspend) Lambda

```kotlin
suspend fun suspendIt(): Unit { }

suspend fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```

```kotlin
suspend fun suspendIt(): Unit { }

suspend fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```

```
1
2

Process finished with exit
code 0
```

```kotlin
suspend fun suspendIt(): Unit { }
```

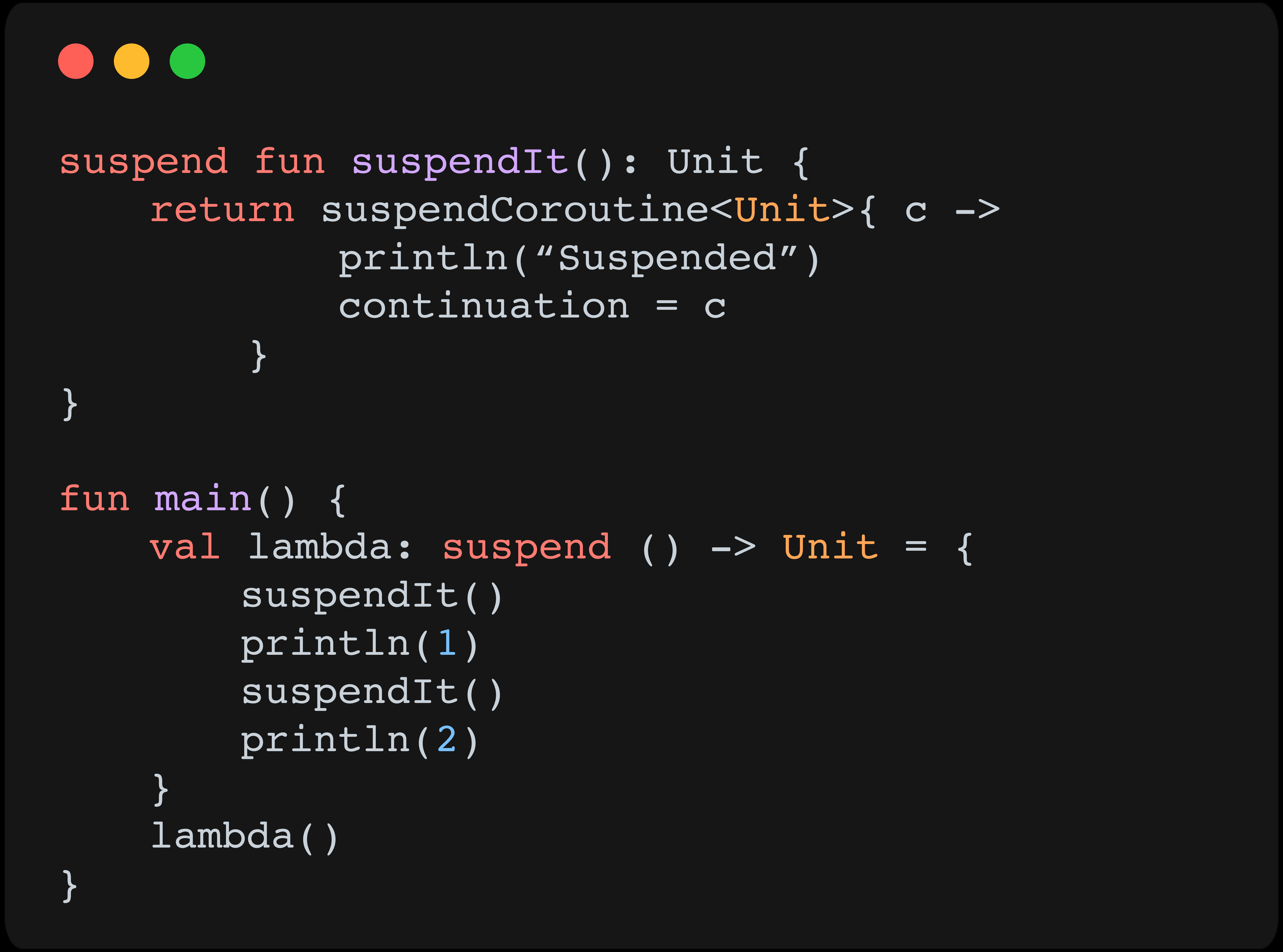
```kotlin
var continuation: Continuation<Unit>? = null

suspend fun suspendIt(): Unit {
    return suspendCoroutine<Unit>{ c ->
            println(“Suspended”)
            continuation = c
        }
}
```

```kotlin
suspend fun suspendIt(): Unit {
    return suspendCoroutine<Unit>{ c ->
            println(“Suspended”)
            continuation = c
        }
}

suspend fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```

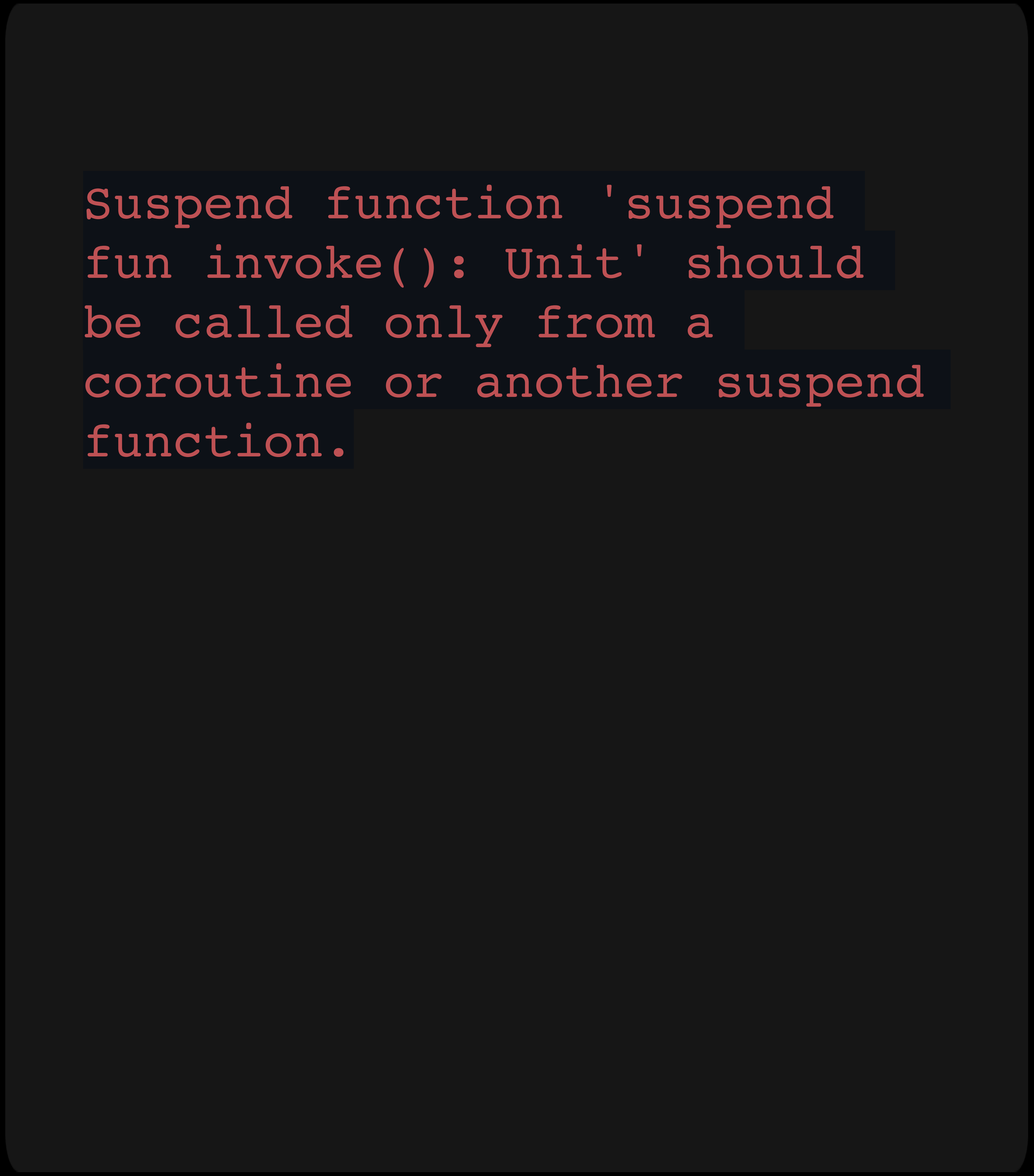
```kotlin
suspend fun suspendIt(): Unit {
    return suspendCoroutine<Unit>{ c ->
            println("Suspended")
            continuation = c
        }
}

suspend fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```

```
Suspended
```

```kotlin
suspend fun suspendIt(): Unit {
    return suspendCoroutine<Unit>{ c ->
            println(“Suspended”)
            continuation = c
        }
}

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```



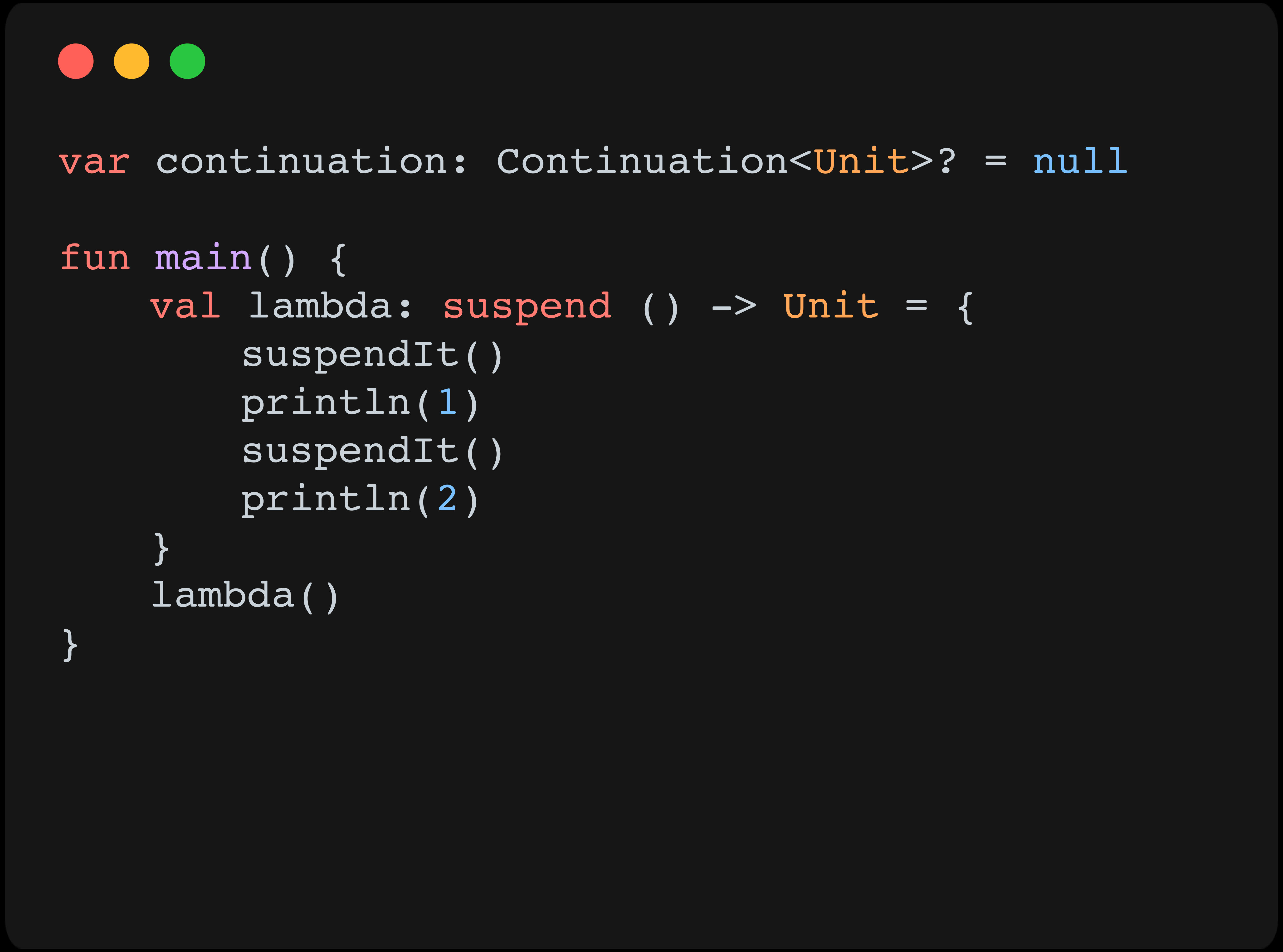
```kotlin
suspend fun suspendIt(): Unit {
    return suspendCoroutine<Unit>{ c ->
            println("Suspended")
            continuation = c
        }
}

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```



```
Suspend function 'suspend
fun invoke(): Unit' should
be called only from a
coroutine or another suspend
function.
```

Kotlin User Groups Seoul

```kotlin
val cont = object : Continuation<Unit> {
    override val context = EmptyCoroutineContext
    override fun resumeWith(result: Result<Unit>) {
        result.getOrThrow()
    }
}

fun runBlocking(func: suspend () -> Unit) {
    func.startCoroutine(cont)
}
```



```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }
    lambda()
}
```

```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
}
```

```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
}
```

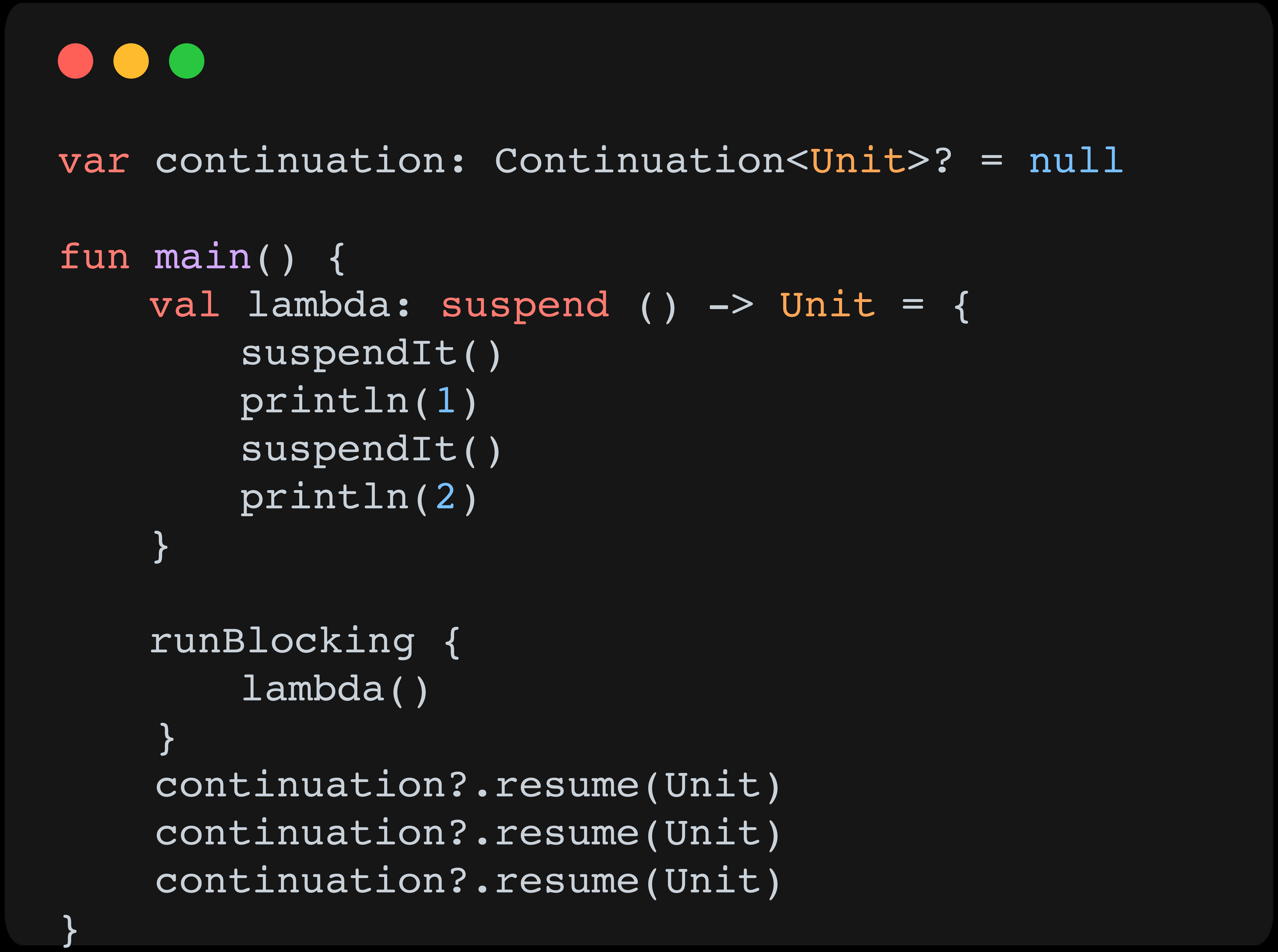
```
Suspended

Process finished with exit
code 0
```

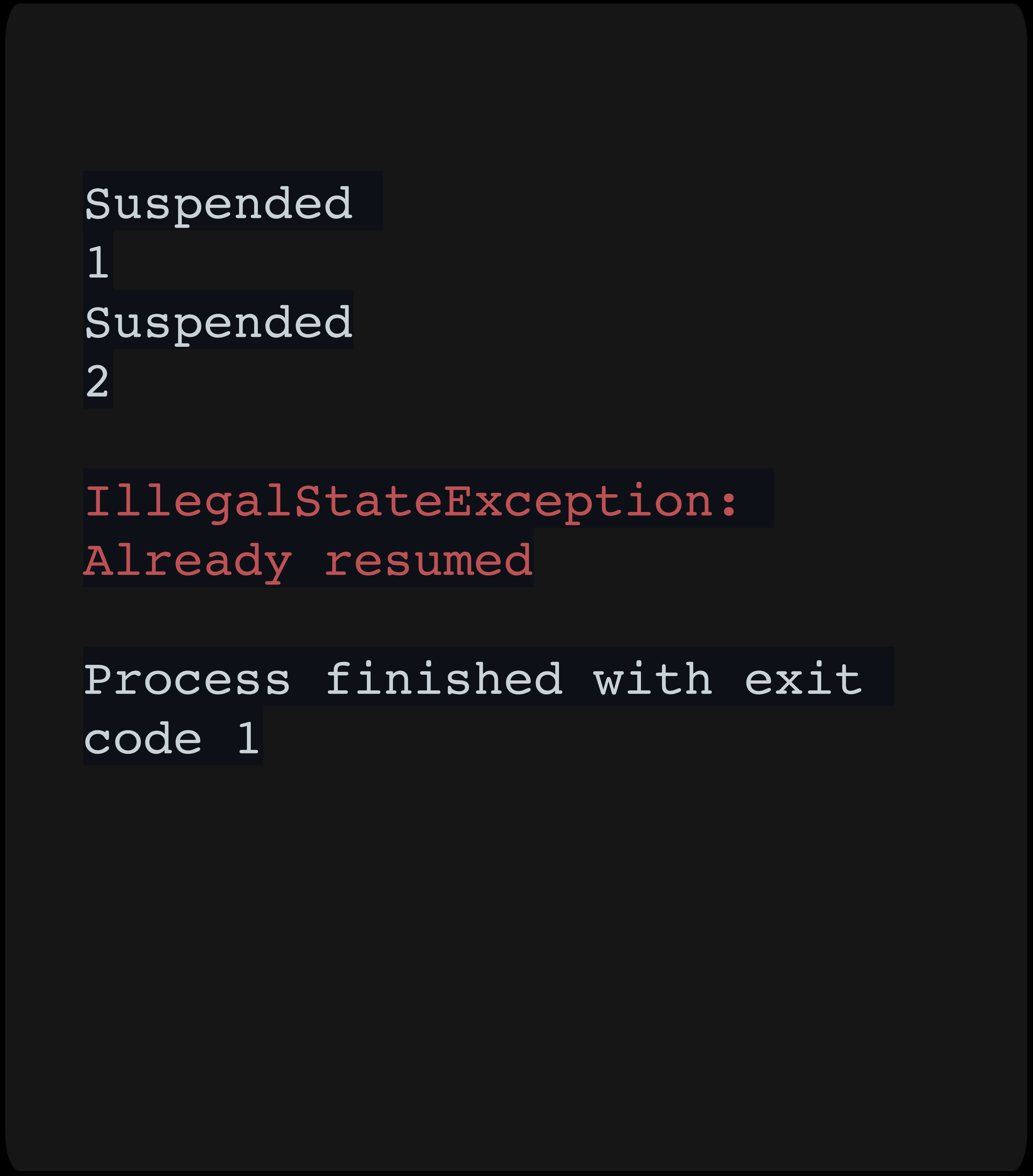
```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
    continuation?.resume(Unit)
}
```

```
Suspended
1
Suspended

Process finished with exit
code 0
```

```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
    continuation?.resume(Unit)
    continuation?.resume(Unit)
}
```

```
Suspended
1
Suspended
2

Process finished with exit
code 0
```


```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
    continuation?.resume(Unit)
    continuation?.resume(Unit)
    continuation?.resume(Unit)
}
```

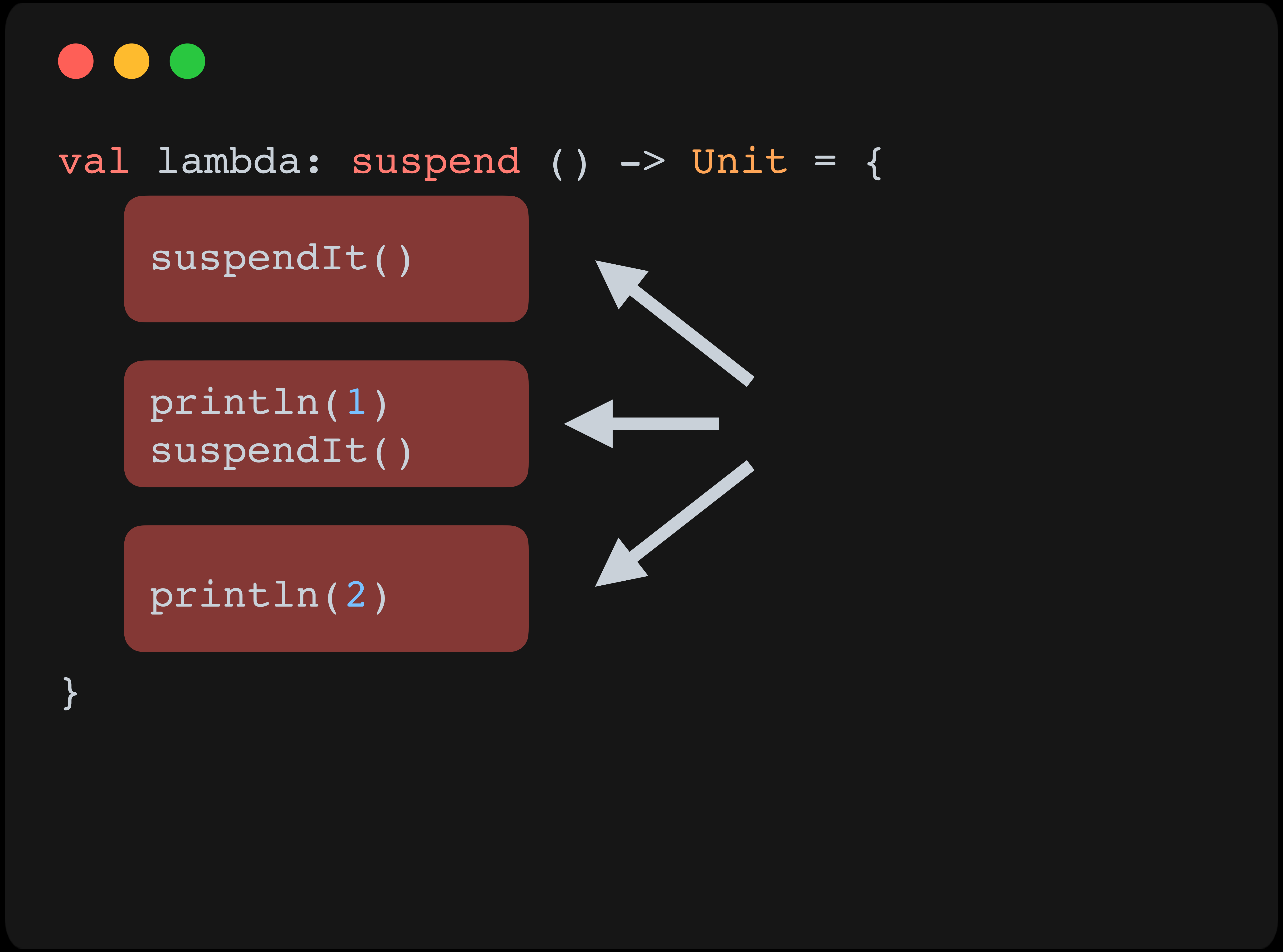

```
Suspended
1
Suspended
2

IllegalStateException:
Already resumed

Process finished with exit
code 1
```

State Machine

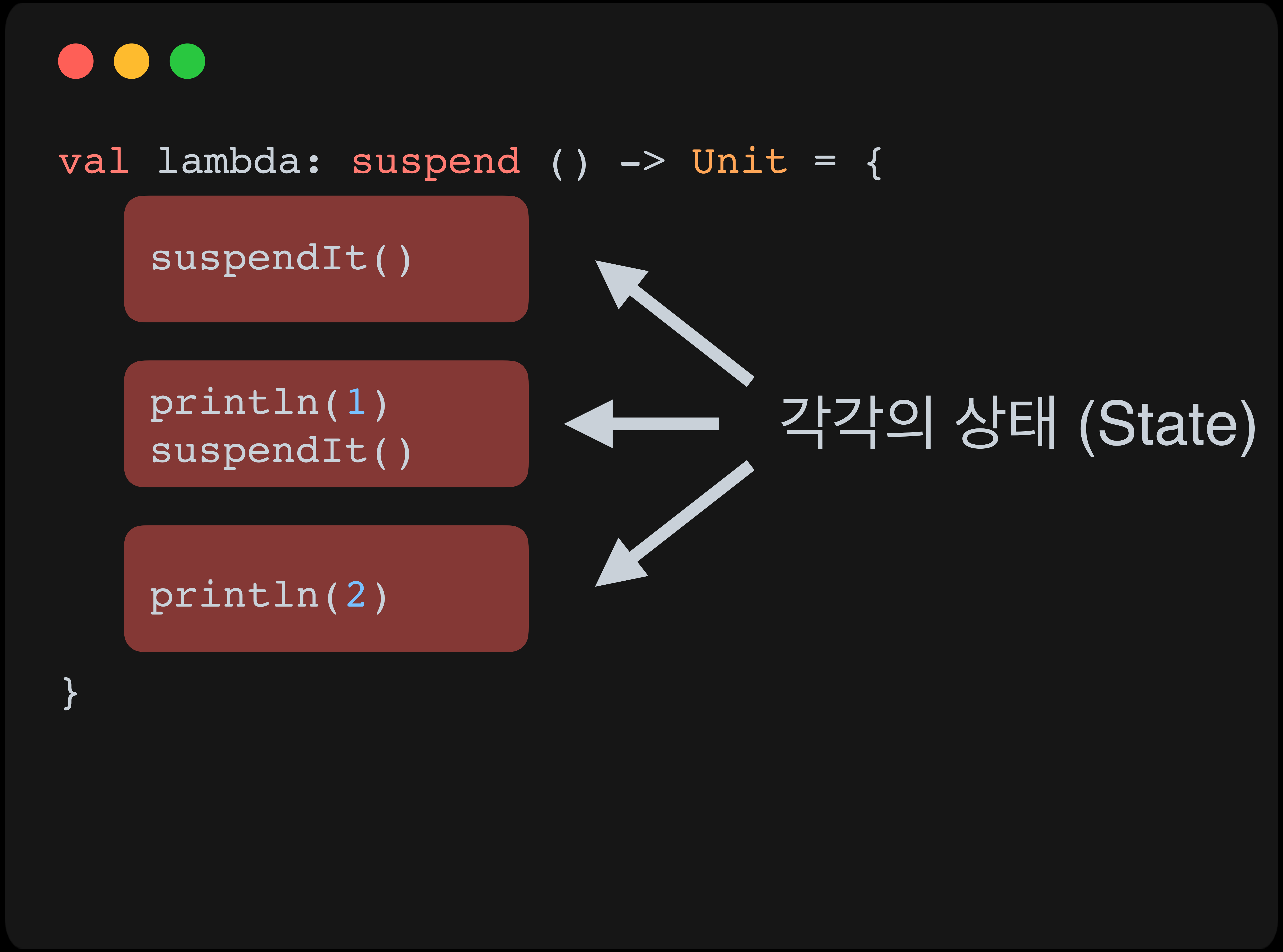
```kotlin
fun main() {
    val lambda: suspend () -> Unit = {
        suspendIt()
        println(1)
        suspendIt()
        println(2)
    }

    runBlocking {
        lambda()
    }
    continuation?.resume(Unit)
    continuation?.resume(Unit)
}
```

```kotlin
val lambda: suspend () -> Unit = {
    suspendIt()
    println(1)
    suspendIt()
    println(2)
}
```

```kotlin
val lambda: suspend () -> Unit = {
    suspendIt()

    println(1)
    suspendIt()

    println(2)
}
```



```kotlin
val lambda: suspend () -> Unit = {
    suspendIt()

    println(1)
    suspendIt()

    println(2)
}
```

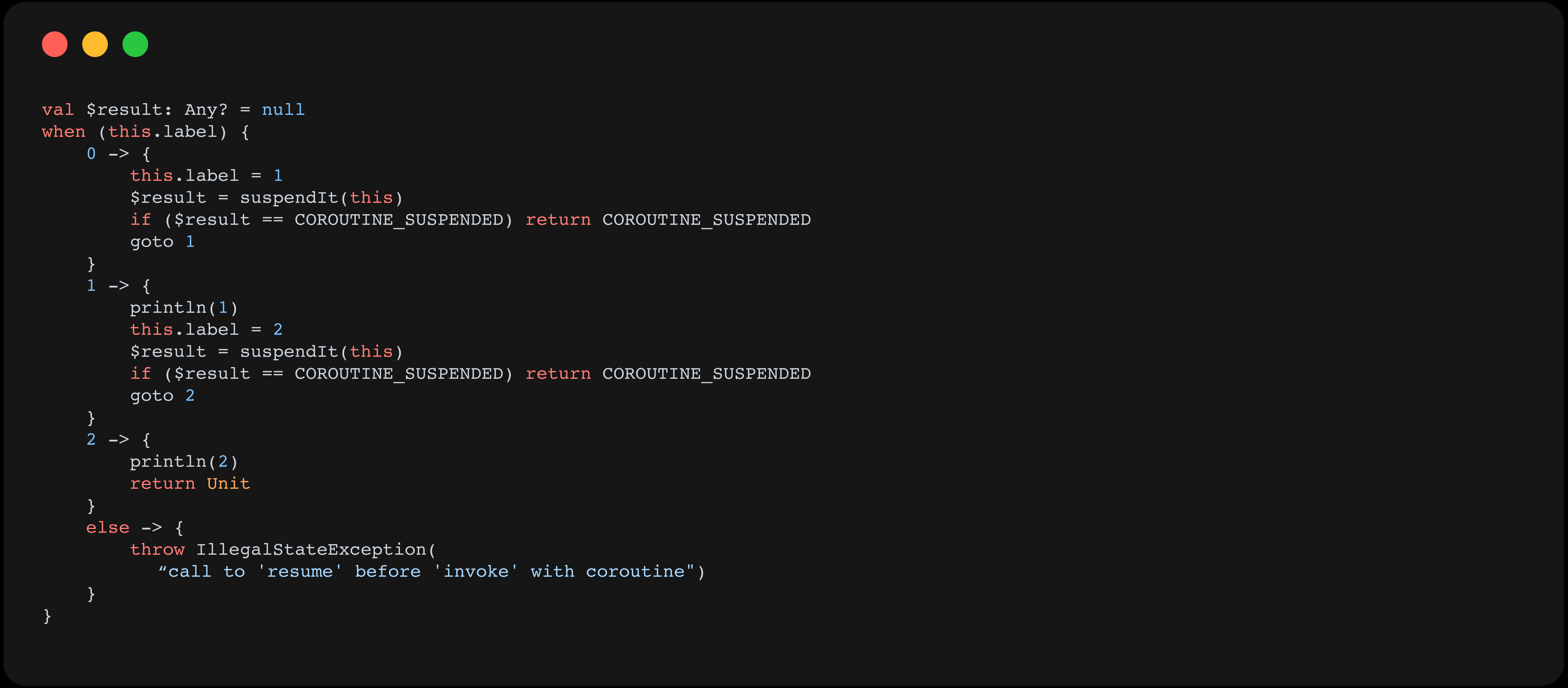


```kotlin
val lambda: suspend () -> Unit = {
    suspendIt()

    println(1)
    suspendIt()

    println(2)
}
```

```kotlin
val lambda: suspend () -> Unit = {
    suspendIt()

    println(1)
    suspendIt()

    println(2)
}
```

invokeSuspend 내부

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
           “call to 'resume' before 'invoke' with coroutine")
    }
}
```



```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
          “call to 'resume' before 'invoke' with coroutine")
    }
}
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

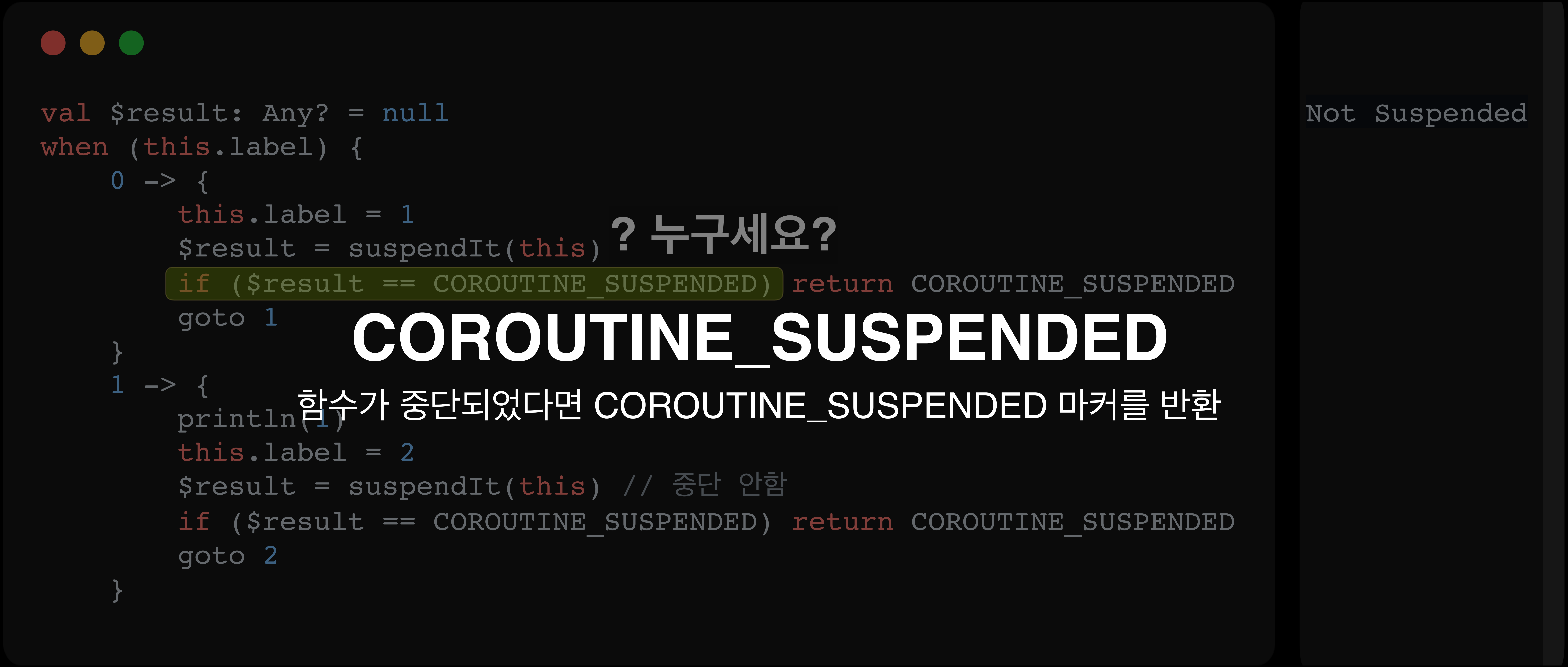
```
Not Suspended
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) ?/ 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) ? 누구세요?
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
```



```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

Not Suspended

? 누구세요?

COROUTINE_SUSPENDED

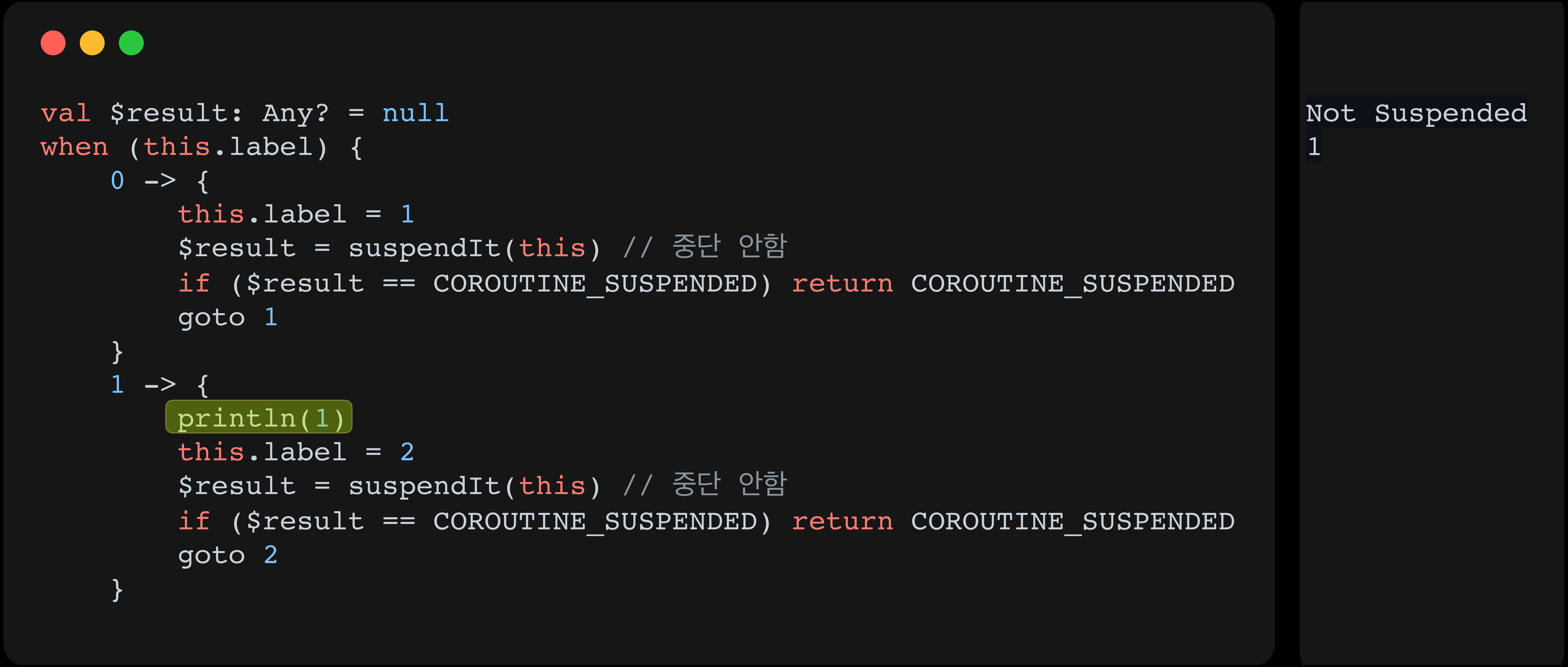
함수가 중단되었다면 COROUTINE_SUSPENDED 마커를 반환

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
```

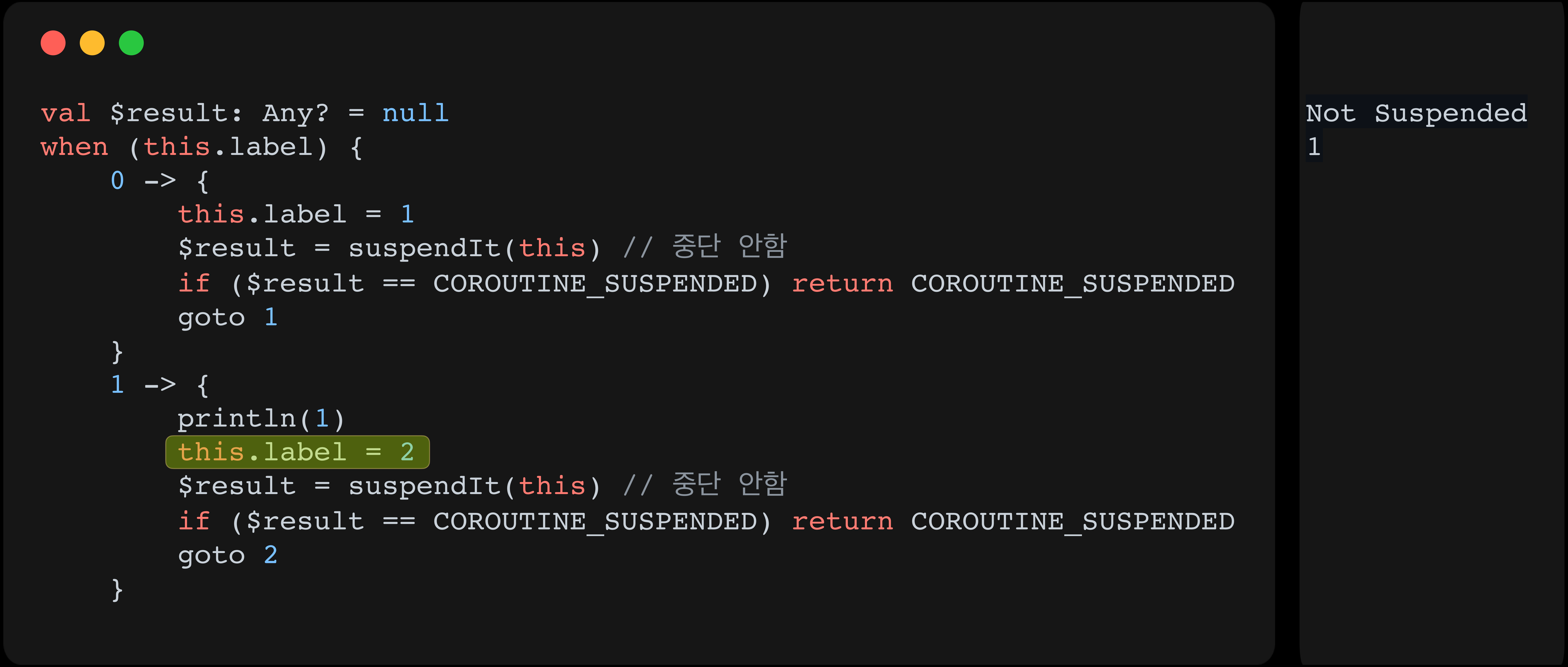
```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
```



```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
1
```

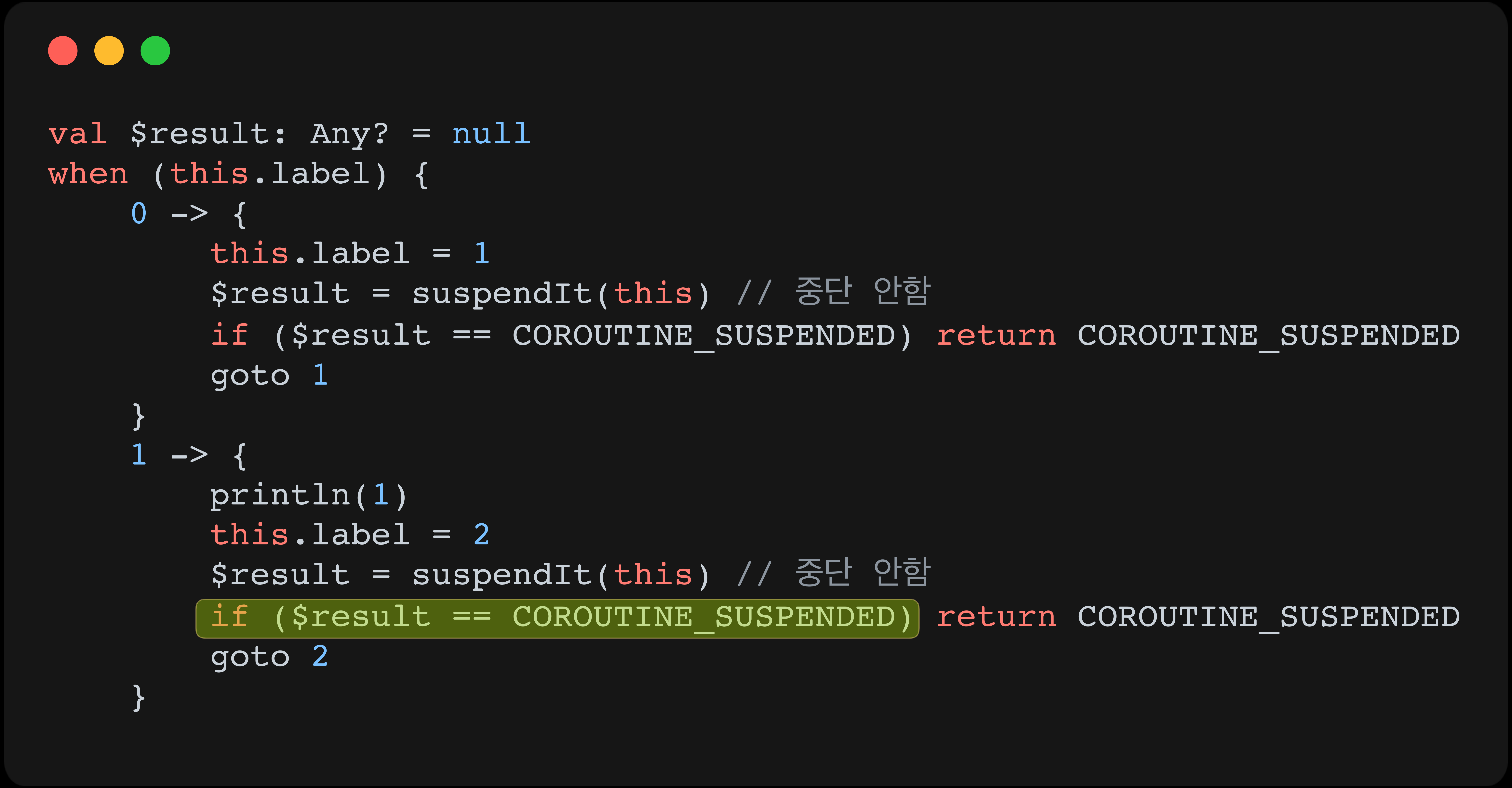


```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

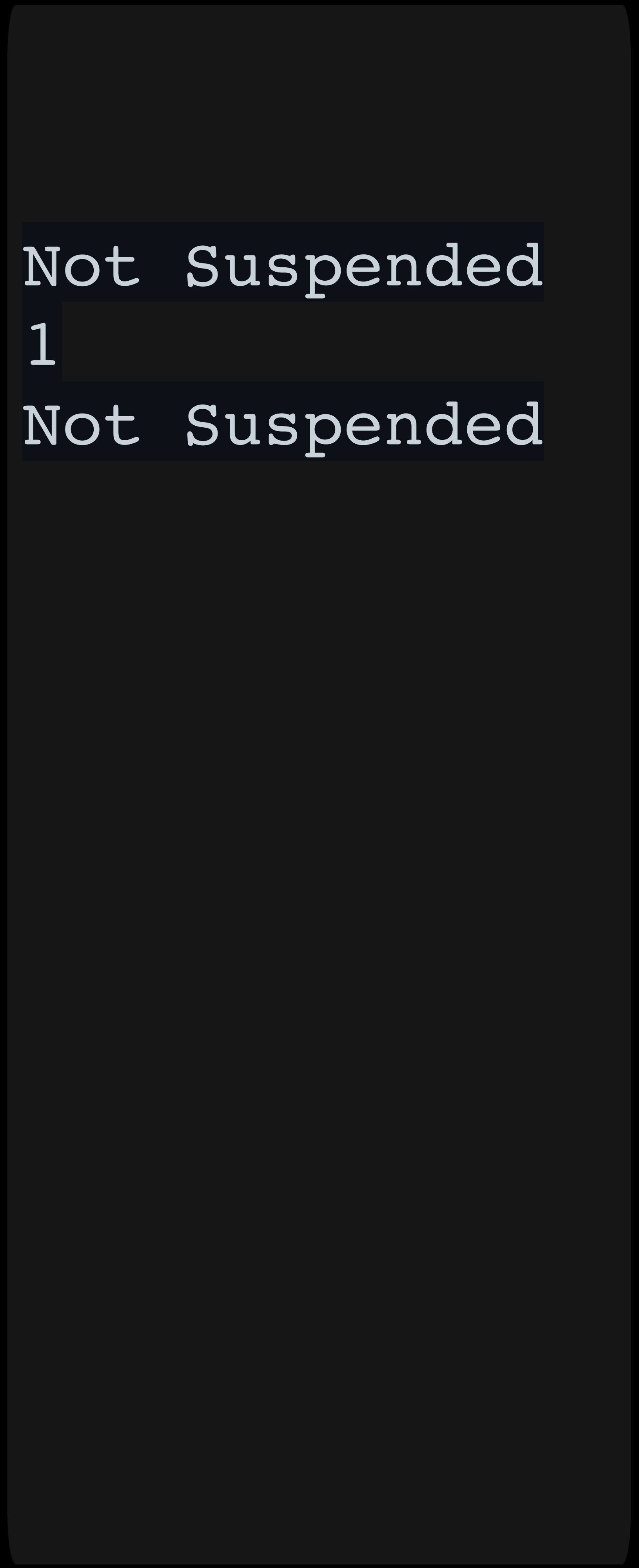
```
Not Suspended
1
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
1
Not Suspended
```



```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```



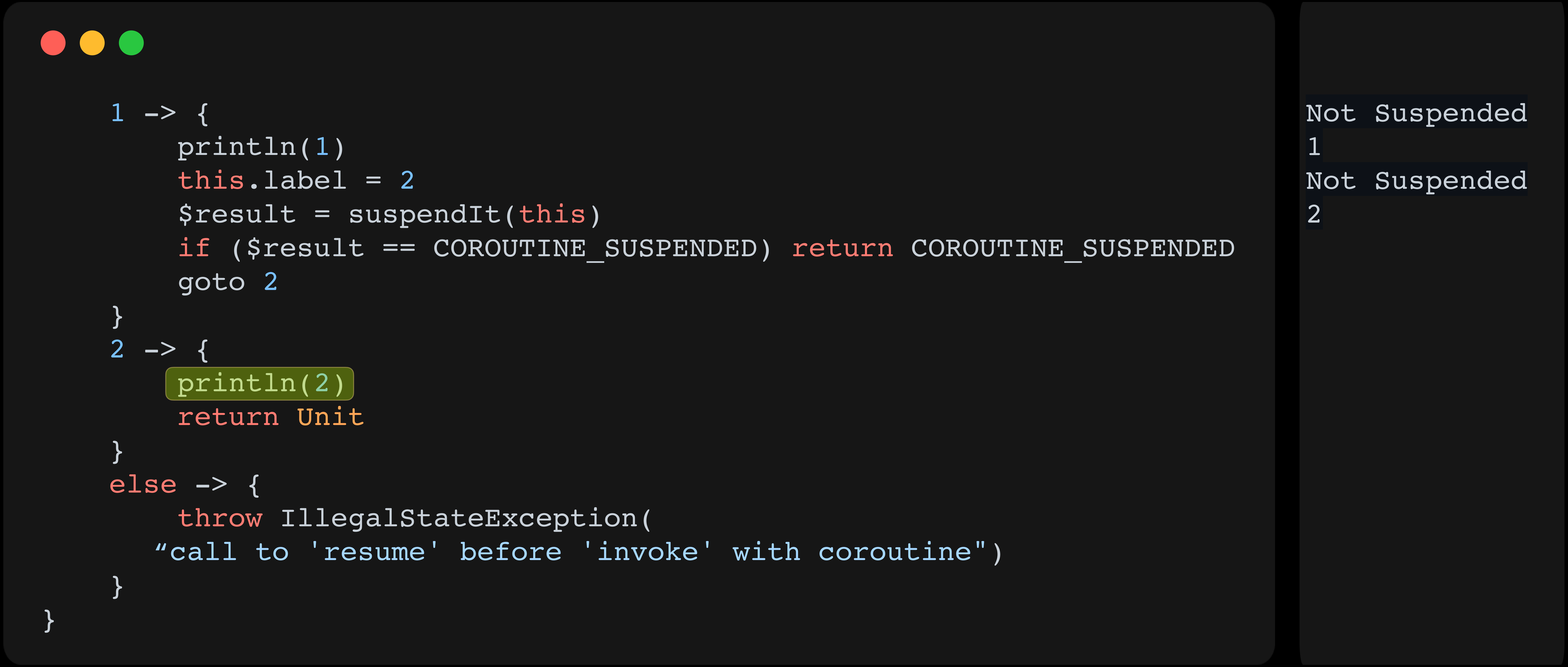
```
Not Suspended
1
Not Suspended
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단 안함
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Not Suspended
1
Not Suspended
```

```kotlin
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
      “call to 'resume' before 'invoke' with coroutine")
    }
}
```

```
Not Suspended
1
Not Suspended
```



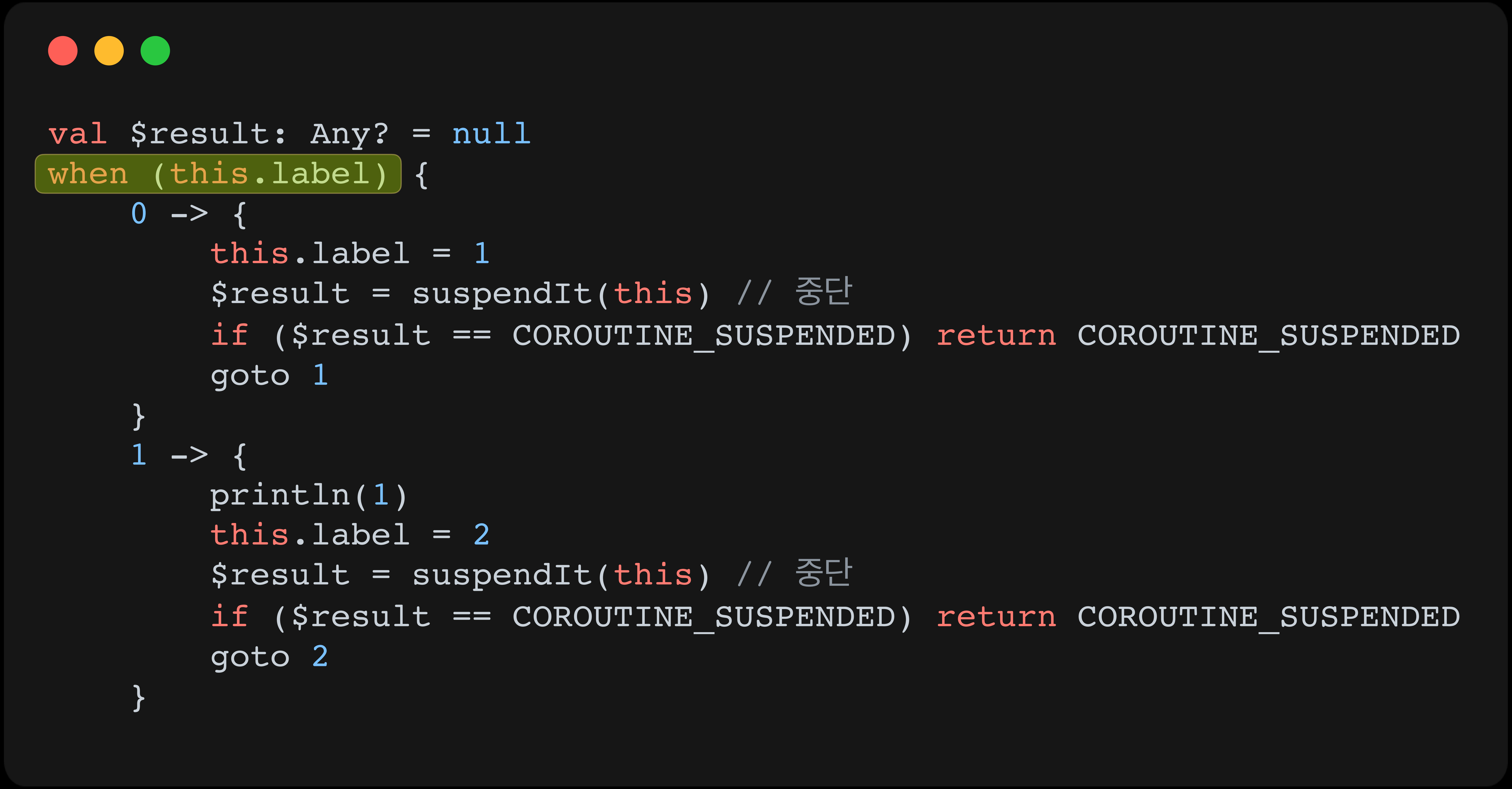
```
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
       “call to 'resume' before 'invoke' with coroutine")
    }
}
```

```
Not Suspended
1
Not Suspended
2
```

```
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
      “call to 'resume' before 'invoke' with coroutine")
    }
}
```

```
Not Suspended
1
Not Suspended
2

exit code 0
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```


```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

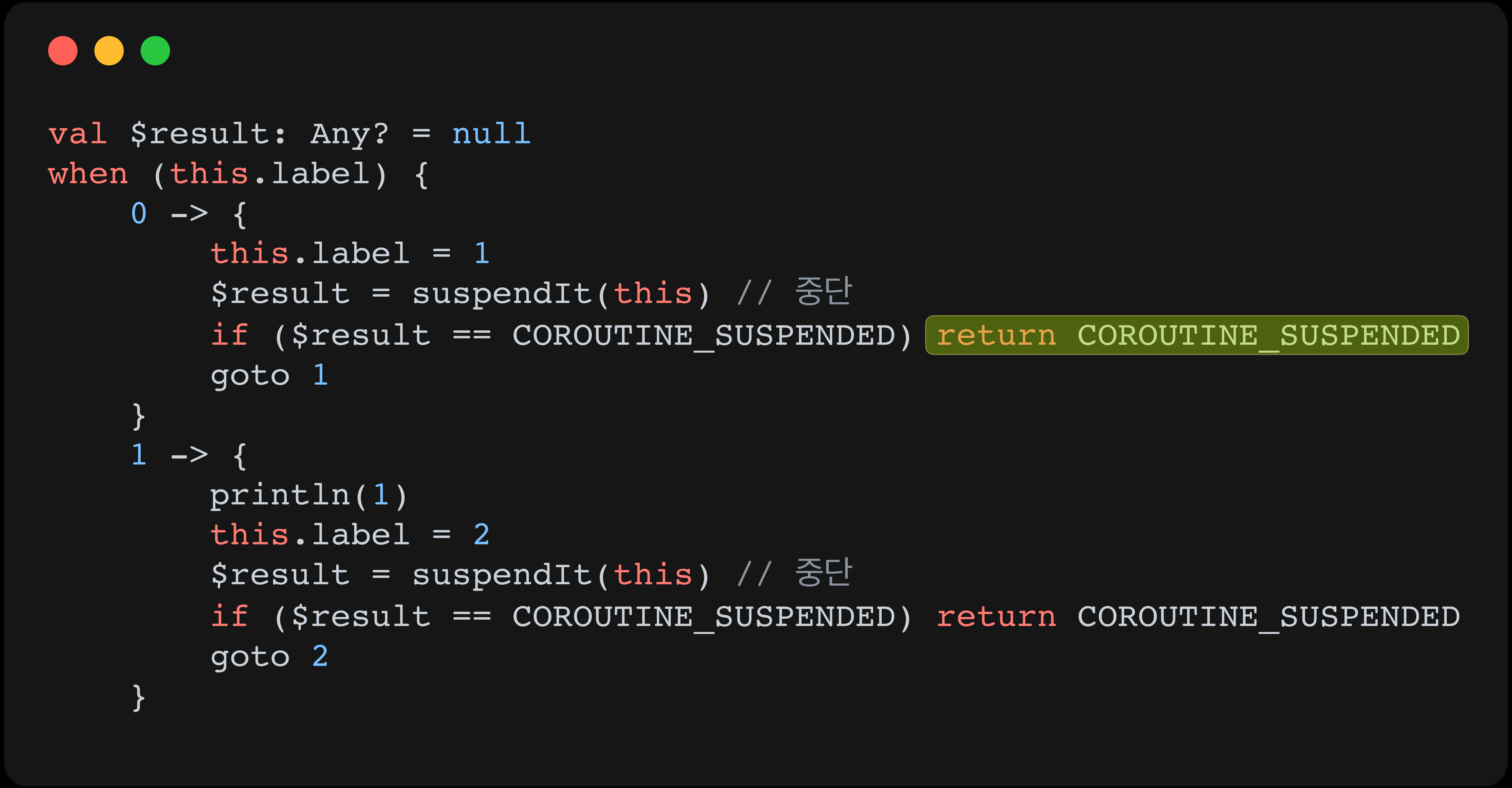
```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Suspended
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Suspended
```

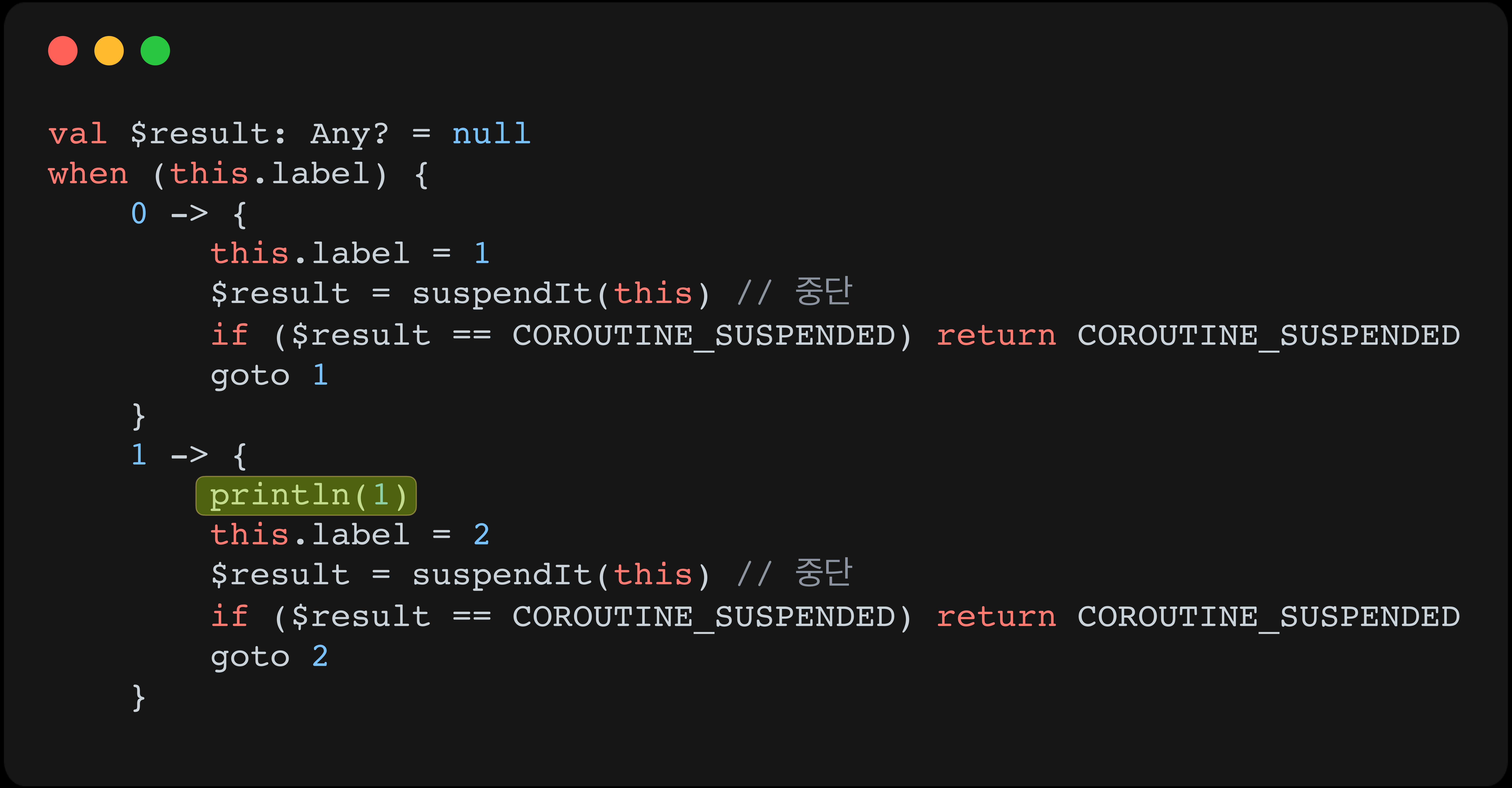


```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

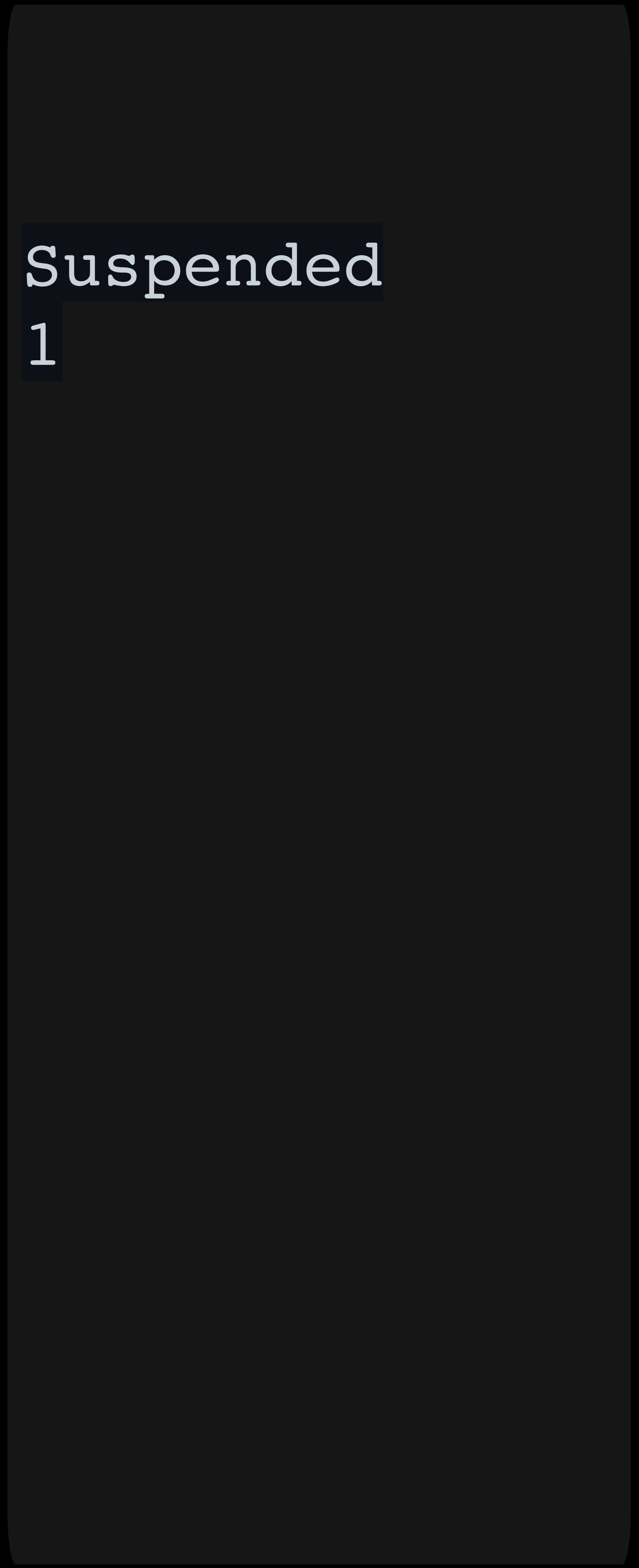
```
Suspended
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Suspended
```



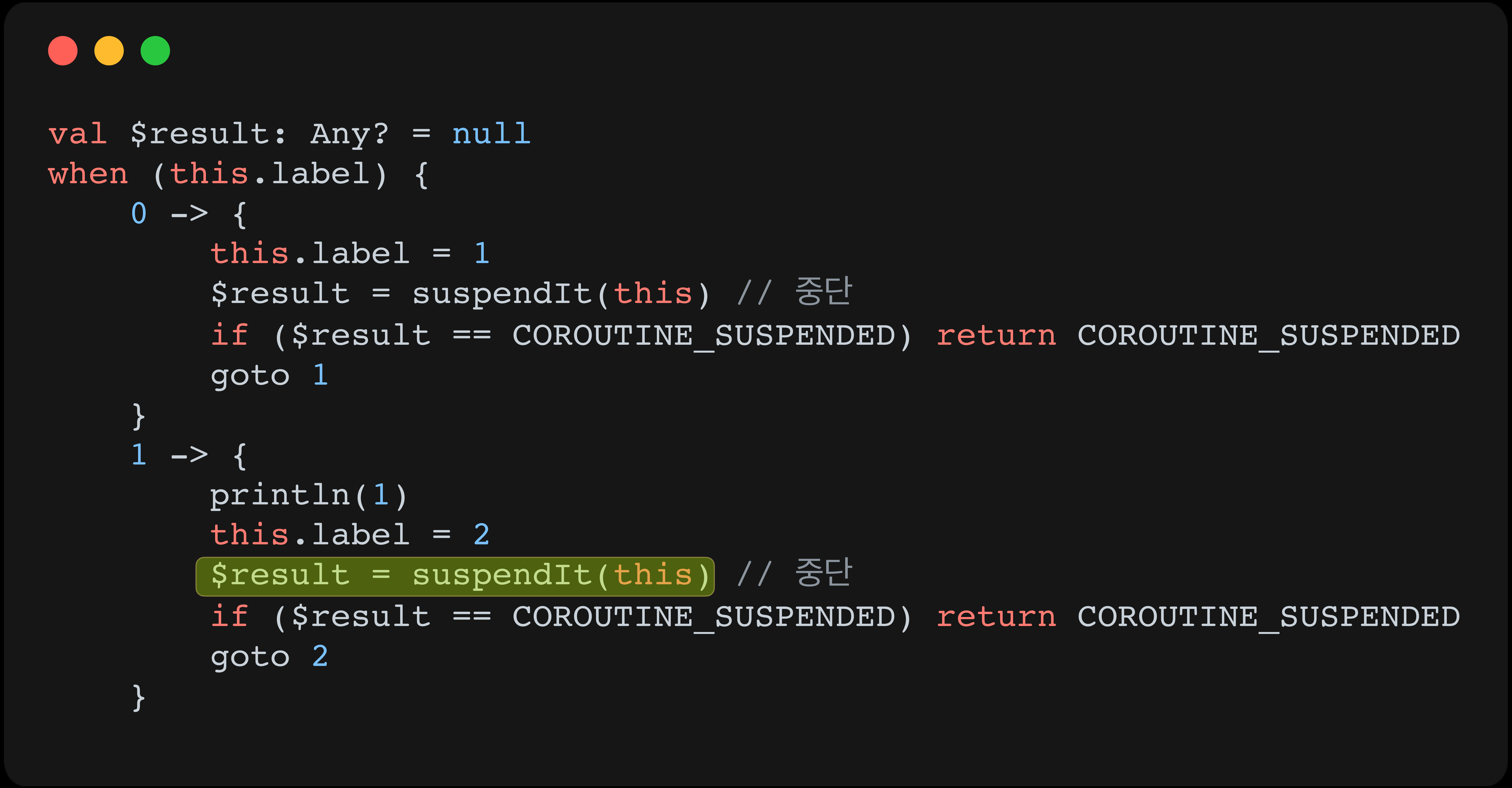
```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```



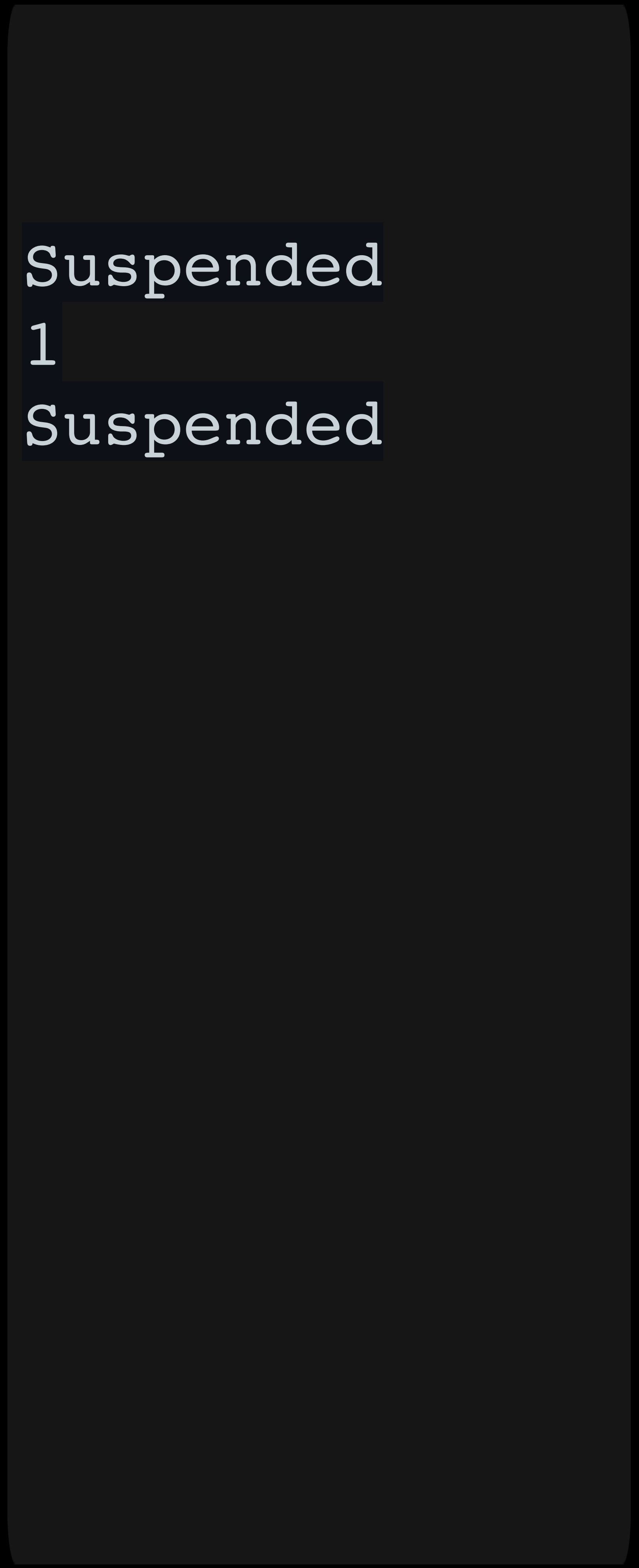
```
Suspended
1
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Suspended
1
```

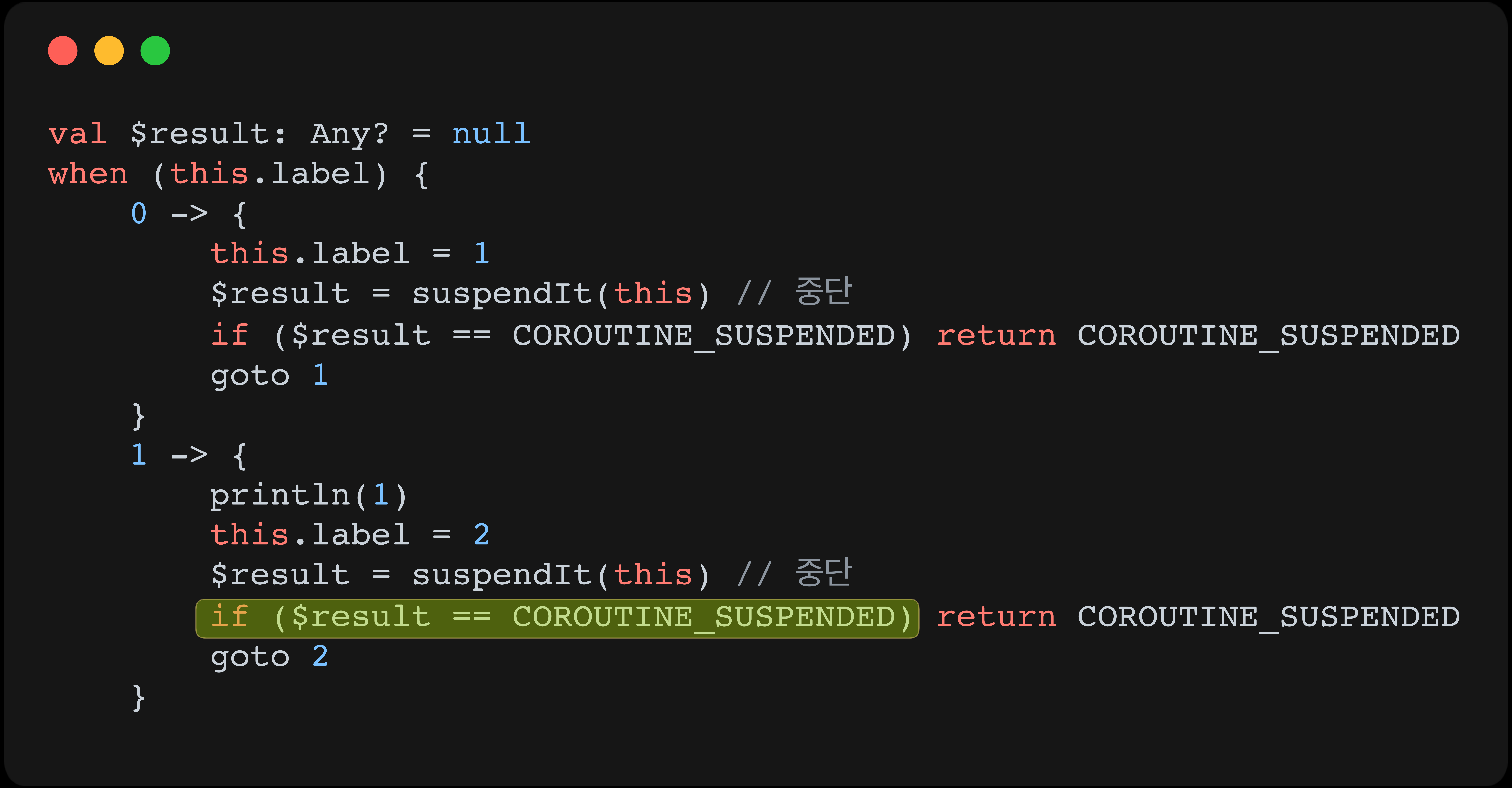


```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```



```
Suspended
1
Suspended
```

Kotlin User Groups Seoul



```
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

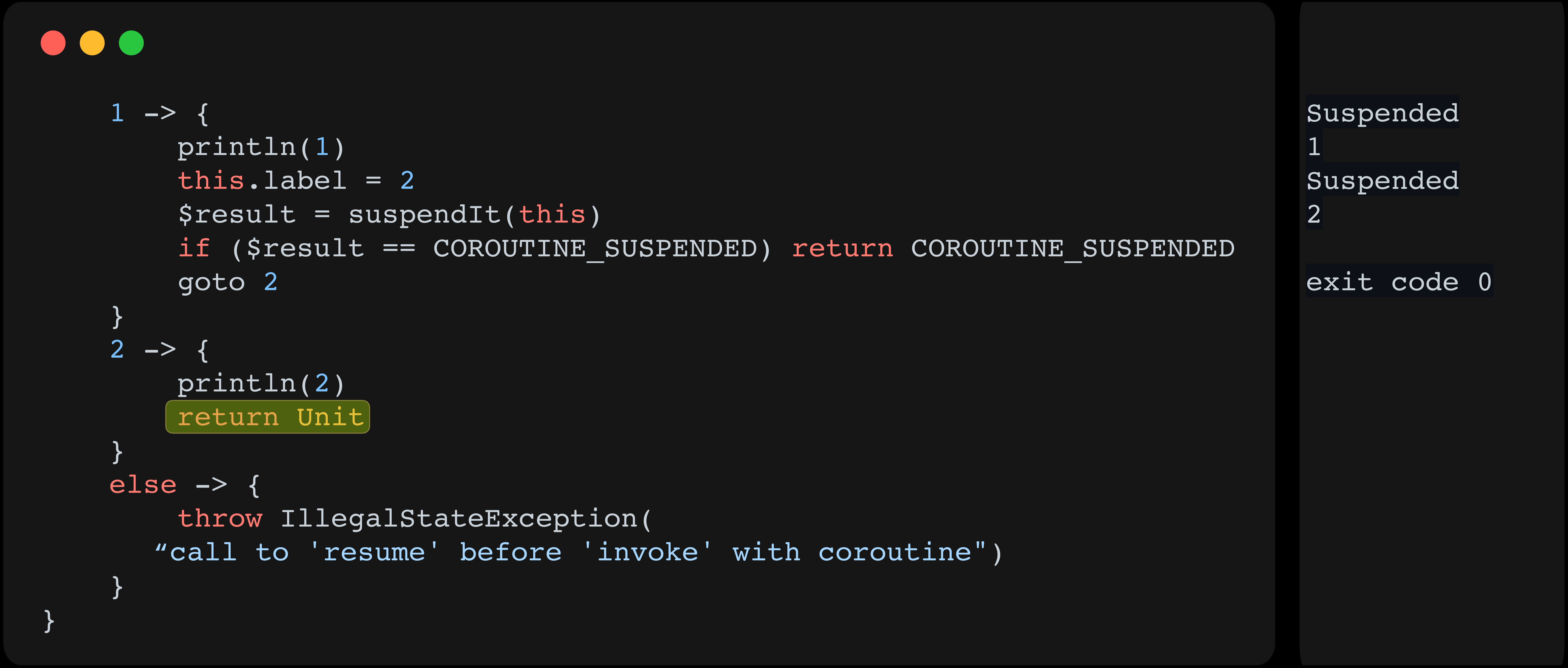
```
Suspended
1
Suspended
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this) // 중단
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

```
Suspended
1
Suspended
```

```
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
      “call to 'resume' before 'invoke' with coroutine")
    }
}
```

```
Suspended
1
Suspended
2
```



```
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
    2 -> {
        println(2)
        return Unit
    }
    else -> {
        throw IllegalStateException(
      "call to 'resume' before 'invoke' with coroutine")
    }
}
```

```
Suspended
1
Suspended
2

exit code 0
```

의문점

COROUTINE_SUSPENDED와 **returnType** 둘 중 하나를
반환 해야한다

의문점

COROUTINE_SUSPENDED와 returnType 둘 중 하나를 반환 해야한다

How?

의문점

COROUTINE_SUSPENDED와 returnType 둘 중 하나를
반환 해야한다

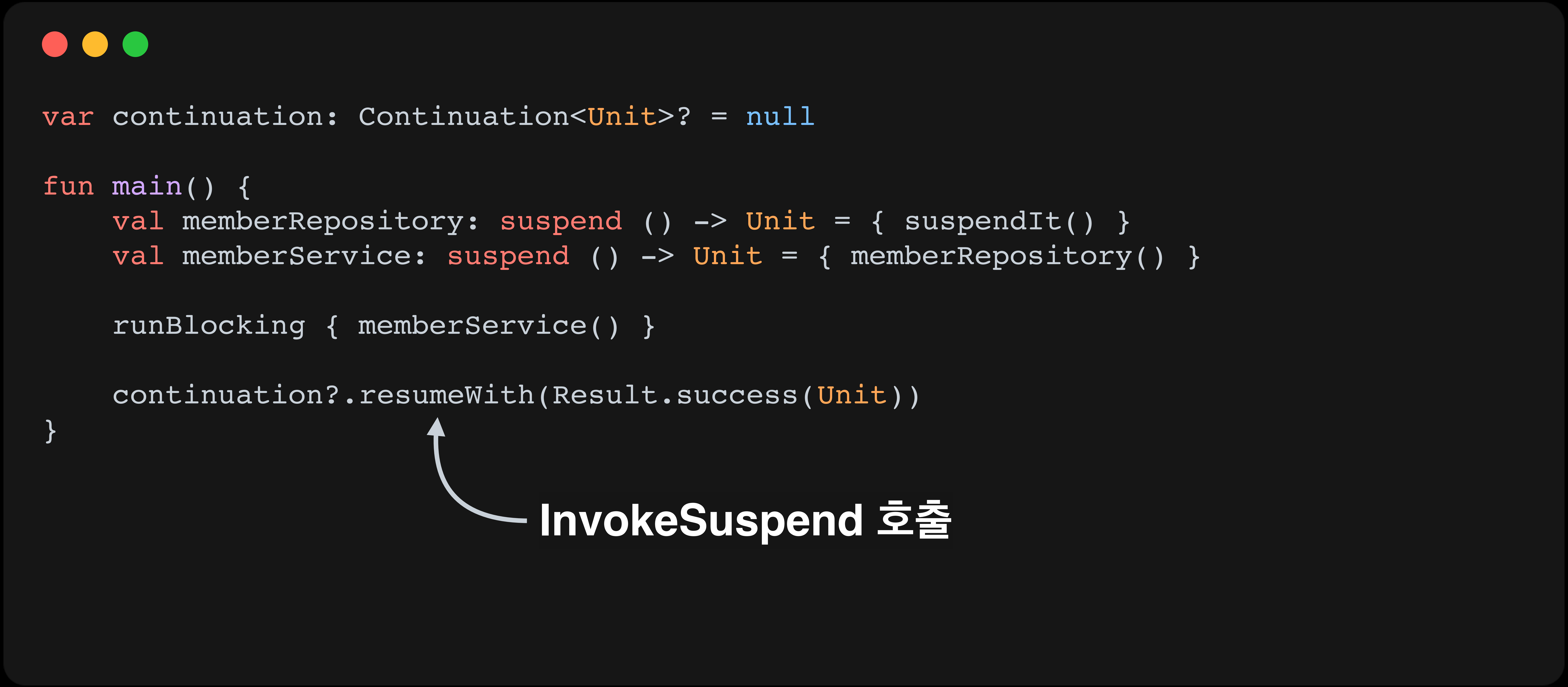
How?

`Any?` 를 반환

Continuation Passing Style

```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val memberRepository: suspend () -> Unit = { suspendIt() }
    val memberService: suspend () -> Unit = { memberRepository() }

    runBlocking { memberService() }

    continuation?.resume(Unit)
}
```

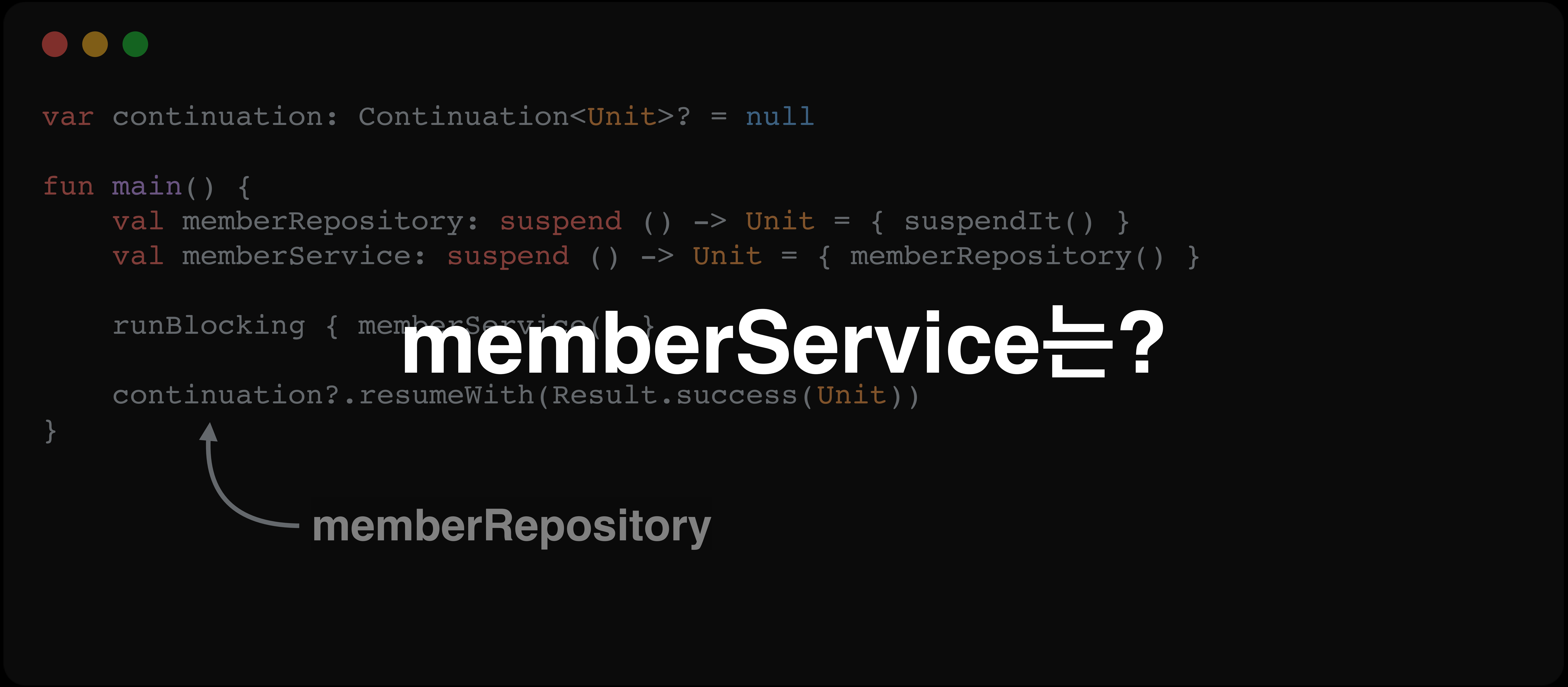
```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val memberRepository: suspend () -> Unit = { suspendIt() }
    val memberService: suspend () -> Unit = { memberRepository() }

    runBlocking { memberService() }

    continuation?.resumeWith(Result.success(Unit))
}
```



```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val memberRepository: suspend () -> Unit = { suspendIt() }
    val memberService: suspend () -> Unit = { memberRepository() }

    runBlocking { memberService() }

    continuation?.resumeWith(Result.success(Unit))
}
```

```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val memberRepository: suspend () -> Unit = { suspendIt() }
    val memberService: suspend () -> Unit = { memberRepository() }

    runBlocking { memberService() }

    continuation?.resumeWith(Result.success(Unit))
}
```

memberRepository



```kotlin
var continuation: Continuation<Unit>? = null

fun main() {
    val memberRepository: suspend () -> Unit = { suspendIt() }
    val memberService: suspend () -> Unit = { memberRepository() }

    runBlocking { memberService() }

    continuation?.resumeWith(Result.success(Unit))
}
```

memberService는?

memberService는 어떻게 실행할까?

memberService는 어떻게 실행할까?

memberRepository가 memberService에 대한 링크를 지니고 있어야한다

memberService는 어떻게 실행할까?

memberRepository가 memberService에 대한 링크를 지니고 있어야한다

즉, memberService를 memberRepository에 전달해야한다

```
fun invoke(): T
```

```kotlin
fun invoke(c: Continuation<Unit>): Any?
fun invoke(c: Continuation<Unit>): Any?
```

이를 Continuation Passing Style이라고 한다

completion

```kotlin
fun invoke(c: Continuation<Unit>): Any?
```

이를 Continuation Passing Style이라고 한다

completion

```
fun invoke(c: Continuation<Unit>): Any?
```

일반 함수에서 suspend 함수를 호출하지 못하는 이유

이를 Continuation Passing Style이라고 한다

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1
    }
    1 -> {
        println(1)
        this.label = 2
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2
    }
```

Local Variables

Local Variables

코루틴이 일시 중단되면, 해당 지역 변수를 저장해야한다

Local Variables

코루틴이 일시 중단되면, 해당 지역 변수를 저장해야한다

그렇지 않으면 코루틴이 다시 재개될 때 변수의 값이 유실된다

Local Variables

코루틴이 일시 중단되면, 해당 지역 변수를 저장해야한다

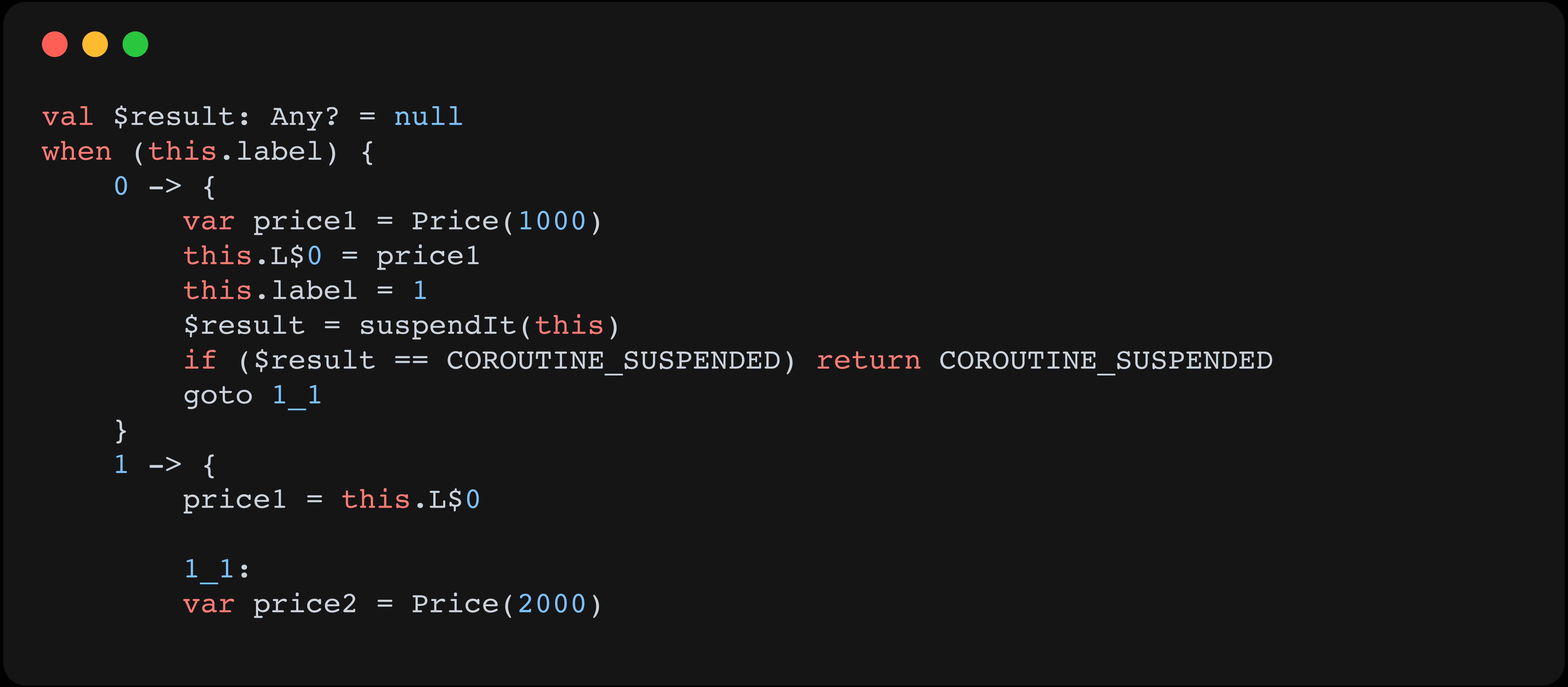
그렇지 않으면 코루틴이 다시 재개될 때 변수의 값이 유실된다

따라서 중단이 발생하기 전에 변수를 저장하고
코루틴이 재개되었을 때, 저장한 변수를 복원해야한다

```kotlin
data class Price(val value: Int) {
    operator fun plus(other: Price): Price {
        return Price(value + other.value)
    }
}

fun main() {
    val lambda: suspend() -> Unit = {
        val price1 = Price(1000)
        suspendIt()
        val price2 = Price(2000)
        suspendIt()
        println(price1 + price2)
    }

}
```

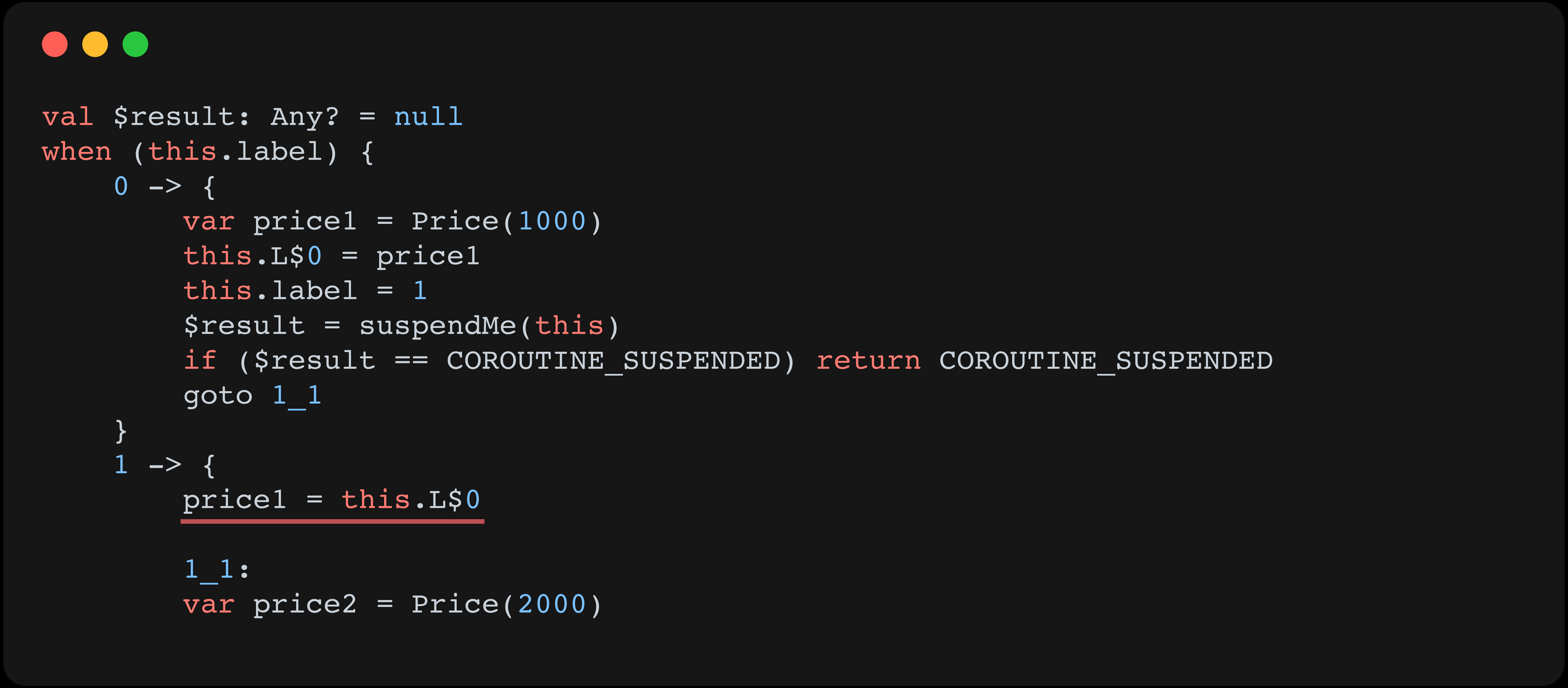
```kotlin
data class Price(val value: Int) {
    operator fun plus(other: Price): Price {
        return Price(value + other.value)
    }
}

fun main() {
    val lambda: suspend() -> Unit = {
        val price1 = Price(1000)
        suspendIt()
        val price2 = Price(2000)
        suspendIt()
        println(price1 + price2)
    }

}
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        var price1 = Price(1000)
        this.L$0 = price1
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1_1
    }
    1 -> {
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
        this.L$0 = null
        this.label = 2
        $result = plus(price1, price2)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2_1
    }
    2 -> {
        2_1:
        println($result)
        return Unit
    }
    else -> {
        throw IllegalStateException("call to 'resume' before 'invoke'
with coroutine")
    }
}
```



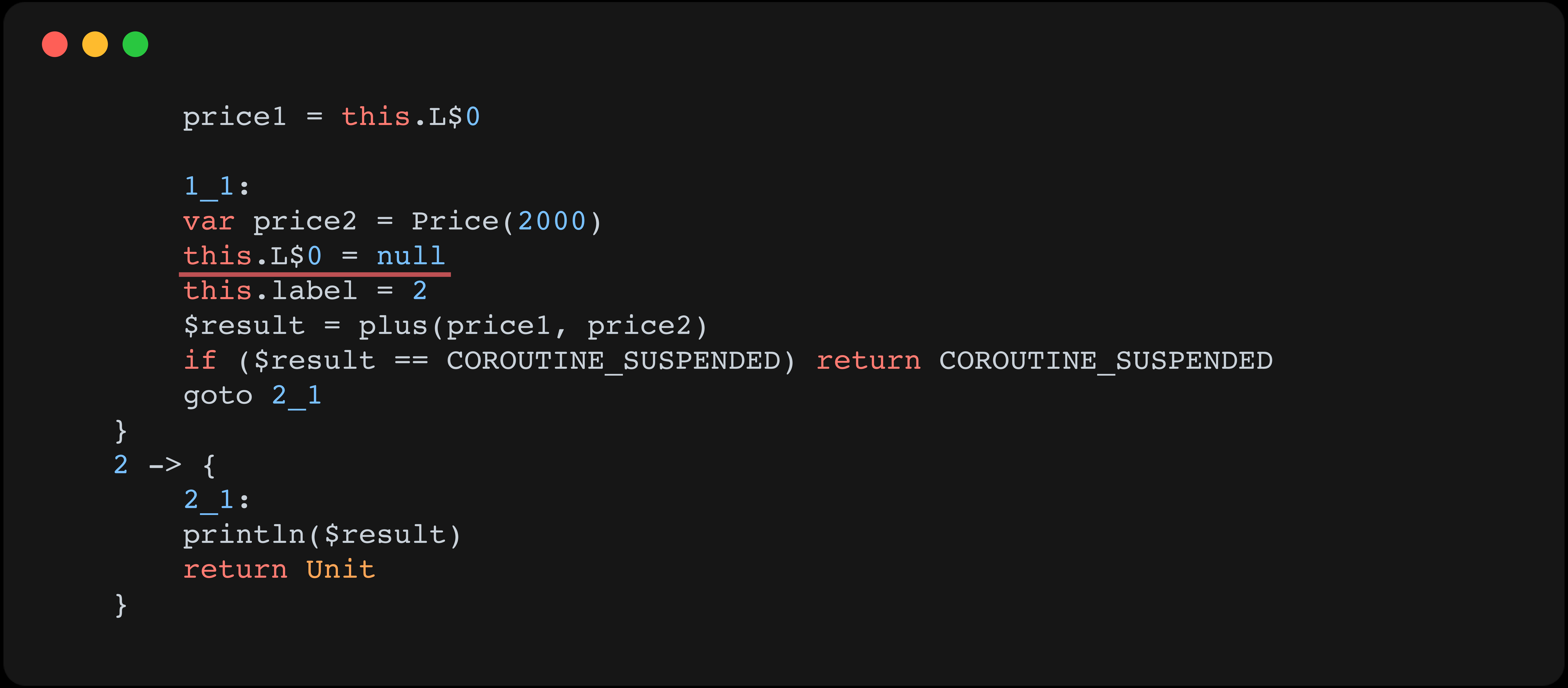
```
val $result: Any? = null
when (this.label) {
    0 -> {
        var price1 = Price(1000)
        this.L$0 = price1
        this.label = 1
        $result = suspendIt(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1_1
    }
    1 -> {
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
```

```
val $result: Any? = null
when (this.label) {
    0 -> {
        var price1 = Price(1000)
        this.L$0 = price1
        this.label = 1
        $result = suspendMe(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1_1
    }
    1 -> {
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
```



```
val $result: Any? = null
when (this.label) {
    0 -> {
        var price1 = Price(1000)
        this.L$0 = price1
        this.label = 1
        $result = suspendMe(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1_1
    }
    1 -> {
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
```

```kotlin
val $result: Any? = null
when (this.label) {
    0 -> {
        var price1 = Price(1000)
        this.L$0 = price1
        this.label = 1
        $result = suspendMe(this)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 1_1
    }
    1 -> {
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
```



```
        price1 = this.L$0

        1_1:
        var price2 = Price(2000)
        this.L$0 = null
        this.label = 2
        $result = plus(price1, price2)
        if ($result == COROUTINE_SUSPENDED) return COROUTINE_SUSPENDED
        goto 2_1
    }
    2 -> {
        2_1:
        println($result)
        return Unit
    }
```

Suspend Function

지금까지 suspend 키워드가 붙은 람다로 진행함

지금까지 suspend 키워드가 붙은 람다로 진행함

Why?

지금까지 suspend 키워드가 붙은 람다로 진행함

Why?

Continuation의 재사용이 불가능함

Suspend Function

Suspend Function

ㄴ 정적 함수일 수 있음

Suspend Function

ㄴ 정적 함수일 수 있음

ㄴ 클래스 내에 여러개 존재 가능

Then?

Then?

모든 함수를 suspend 람다로 변환

JVM은 왜 이 방법을 사용하지 않을까?

JVM은 왜 이 방법을 사용하지 않을까?

스택 트레이스(Stack Trace)의 가독성

함수 본문을 Lambda로 이동한 경우

```
suspendFun1$1.invokeSuspend
suspendFun2$1.invokeSuspend
suspendFun3$1.invokeSuspend
suspendFun4$1.invokeSuspend
suspendFun5$1.invokeSuspend
```

우리가 원하는 형태

```
suspendFun1
suspendFun2
suspendFun3
suspendFun4
suspendFun5
```

How?

Suspend 함수의 본문을 람다로 이동하는 대신, 함수 내부에 그대로 유지하고 상태기계를 그 안에 직접 구축하는 방식을 채택

감사합니다.

QnA

linkedin.com/in/chanwoo040531/