# KOTLIN COROUTINE - TO'LIQ QO'LLANMA (O'ZBEK TILIDA)

===============================================================

## 1-BO'LIM: COROUTINE NIMA?

Coroutine - bu Kotlin tilida asinxron (async) kod yozish uchun ishlatiladigan mexanizm. Oddiy qilib aytganda, bu "yengil og'irlikdagi thread" (light-weight thread).

--- ODDIY MISOLDA ---

❌MUAMMO (Coroutine'siz):

```
Button(onClick = {
// Bu kod UI thread'da ishlaydi
downloadLargeFile() // 10 soniya davom etadi
updateUI() // Bu yerga 10 soniyadan keyin yetib keladi
// Dastur 10 soniya "muzlab" turadi! 😱
})
```

✅YECHIM (Coroutine bilan):

```
Button(onClick = {
coroutineScope.launch {
downloadLargeFile() // Orqa fonda ishlaydi
updateUI() // Tugagach UI yangilanadi
// Dastur muzlamaydi! 😊
}
})
```

--- REAL HAYOTDAN MISOL: McDONALD'S ---

Oddiy usul (Sinxron):

1. Siz buyurtma berdingiz 🍔
2. Ishchi sizning buyurtmangizni tayyorlayapti
3. Orqadagi 100 ta odam kutib turishadi 😤
4. Sizniki tayyor bo'ldi, keyingisiga o'tiladi

Coroutine usuli (Asinxron):

1. Siz buyurtma berdingiz 🍔
2. Sizga raqam berildi (Token)
3. Keyingi odamlar ham buyurtma berishadi ✅
4. Sizniki tayyor bo'lganda chaqirishadi 🔔

--- SNACKBAR NIMA UCHUN COROUTINE ISHLATADI? ---

myCoroutineScope.launch {
val result = mySnackbarHostState.showSnackbar(...)
// Bu yerda Snackbar ko'rsatilguncha kutadi
// Lekin UI muzlamaydi!
}

SABABI:

- showSnackbar() funksiyasi suspend function (to'xtatiluvchi funksiya)
- U Snackbar yopilguncha kutadi (2-3 soniya yoki Indefinite)
- Agar coroutine bo'lmasa, dastur muzlab qolardi

--- THREAD VS COROUTINE ---

| Thread | Coroutine |
|---|---|
| Og'ir (heavy) | Yengil (light) |
| Ko'p xotira oladi | Kam xotira oladi |
| Yaratish qimmat | Yaratish arzon |
| 1000 ta thread = muammo | 100,000 ta coroutine = OK |

**==================================================================**

## 2-BO'LIM: COROUTINE QAYERLARDA ISHLATILADI?

Coroutine Android developmentda ENG KO'P ishlatiladigan narsalardan biri!

--- 1. API DAN MA'LUMOT OLISH (Eng keng tarqalgan) ---

```
Button(onClick = {
coroutineScope.launch {
try {
val users = api.getUsers() // Internet'dan ma'lumot
userList.value = users // UI'ni yangilash
} catch (e: Exception) {
Toast.makeText(context, "Xatolik!", Toast.LENGTH_SHORT).show()
}
}
})
```

--- 2. MA'LUMOTLAR BAZASI BILAN ISHLASH (Room Database) ---

```
// Ma'lumot qo'shish
viewModelScope.launch {
database.userDao().insert(newUser)
}
```

```
// Ma'lumot o'qish
viewModelScope.launch {
val allUsers = database.userDao().getAllUsers()
_userListState.value = allUsers
}
```

--- 3. FAYL YUKLASH/YUKLASH (Download/Upload) ---

```
Button(onClick = { coroutineScope.launch { downloadFile("https://example.com/file.pdf")
Toast.makeText(context, "Yuklash tugadi!", Toast.LENGTH_SHORT).show() } })
```

```
suspend fun downloadFile(url: String) {
// Fayl yuklash logikasi
delay(5000) // 5 soniya kutish (misol)
}
```

--- 4. BIR NECHA AMALLARNI KETMA-KET BAJARISH ---

```
coroutineScope.launch {
// 1. Login
val token = loginUser(email, password)
```

```
// 2. Token bilan profil olish
val profile = getUserProfile(token)

// 3. UI'ni yangilash
_userState.value = profile
```

}

--- 5. PARALLEL ISHLARNI BAJARISH (async/await) ---

coroutineScope.launch {
// Bir vaqtning o'zida 3 ta so'rov
val profile = async { api.getProfile() }
val posts = async { api.getPosts() }
val followers = async { api.getFollowers() }

```
// Hammasini kutish
showUserData(
    profile.await(),
    posts.await(),
    followers.await()
)
```

}

--- 6. VAQT BILAN BOG'LIQ OPERATSIYALAR ---

// Timer
coroutineScope.launch {
repeat(10) { i ->
delay(1000) // Har 1 soniyada
timerText.value = "${10 - i} soniya qoldi"
}
Toast.makeText(context, "Vaqt tugadi!", Toast.LENGTH_SHORT).show()
}

// Countdown
coroutineScope.launch {
var count = 60
while (count > 0) {
delay(1000)
count--
countdownText.value = count.toString()
}
}

--- 7. VIEWMODEL'DA ISHLATISH (Eng to'g'ri usul) ---

class MyViewModel : ViewModel() {

```
fun loadData() {
    viewModelScope.launch {
        _isLoading.value = true

        val data = repository.getData()
        _dataState.value = data

        _isLoading.value = false
    }
}

fun saveData(data: User) {
    viewModelScope.launch {
        repository.save(data)
        Toast.makeText(context, "Saqlandi!", Toast.LENGTH_SHORT).show()
    }
}
```

}

--- 8. DISPATCHER'LAR (Qayerda ishlashini tanlash) ---

```
coroutineScope.launch {
// Main (UI) thread'da
withContext(Dispatchers.Main) {
updateUI()
}
```

```
// Orqa fon thread'da (og'ir hisob-kitoblar)
withContext(Dispatchers.Default) {
    val result = heavyCalculation()
}

// I/O operatsiyalari (network, database, fayl)
withContext(Dispatchers.IO) {
    val data = database.query()
}
```

```
}
```

--- REAL LOYIHADA MISOL ---

```
class LoginViewModel : ViewModel() {
```

```
fun login(email: String, password: String) {
    viewModelScope.launch {
        try {
            // 1. Loading ko'rsatish
            _isLoading.value = true

            // 2. API'ga so'rov
            val response = authRepository.login(email, password)

            // 3. Token saqlash
            dataStore.saveToken(response.token)

            // 4. Profil olish
            val profile = userRepository.getProfile(response.token)

            // 5. Ma'lumotlar bazasiga saqlash
            database.userDao().insert(profile)

            // 6. Navigation
            _navigateToHome.value = true

        } catch (e: Exception) {
            _errorMessage.value = "Login xatolik: ${e.message}"
        } finally {
            _isLoading.value = false
        }
    }
}
```

}

--- COROUTINE QACHON ISHLATILADI ---

✅Network (API) so'rovlar
✅Database operatsiyalari
✅Fayl operatsiyalari
✅Og'ir hisob-kitoblar
✅Vaqt talab qiladigan har qanday ish
✅UI'ni muzlatmaslik kerak bo'lgan operatsiyalar

QOIDA: Agar operatsiya 16ms dan ko'p vaqt olsa (1 frame), coroutine ishlatish kerak!

**================================================================**

## 3-BO'LIM: SUSPEND FUNCTION NIMA?

suspend = "to'xtatiluvchi" yoki "kutish mumkin"

Bu maxsus funksiya bo'lib, uni faqat COROUTINE ICHIDA chaqirish mumkin.

--- ODDIY VS SUSPEND FUNKSIYA ---

// ❌Oddiy funksiya - Coroutine'siz ishlaydi
fun normalFunction() {
println("Salom")
}

// ✅Suspend funksiya - Faqat coroutine ichida ishlaydi
suspend fun suspendFunction() {
delay(1000)
println("Salom")
}

--- MISOL ---

suspend fun downloadData() {
delay(2000) // 2 soniya kutadi
println("Ma'lumot yuklandi")
}

Nima bo'lyapti:

1. suspend fun - Bu funksiyani faqat coroutine ichida chaqirish mumkin
2. delay(2000) - 2000 millisekund (2 soniya) kutish
3. println(...) - 2 soniyadan keyin "Ma'lumot yuklandi" yoziladi

--- QANDAY CHAQIRILADI ---

// ❌XATO - Oddiy kod ichida ishlamaydi
fun myFunction() {
downloadData() // XATO!
}

// ✅TO'G'RI - Coroutine ichida
fun myFunction() {
coroutineScope.launch {
downloadData() // TO'G'RI!
}
}

--- delay() VS Thread.sleep() ---

❌Thread.sleep() - Yomon usul:

fun badDownload() {
Thread.sleep(2000) // 2 soniya
// ⚠️Muammo: Butun thread to'xtaydi!
// UI muzlab qoladi! 😱
println("Ma'lumot yuklandi")
}

✅delay() - Yaxshi usul:

suspend fun goodDownload() {
delay(2000) // 2 soniya
// ✅Faqat bu coroutine to'xtaydi
// UI ishlashda davom etadi! 😊
println("Ma'lumot yuklandi")
}

--- REAL MISOL ---

@Composable
fun MyScreen() {
val scope = rememberCoroutineScope()
var status by remember { mutableStateOf("Bosing") }

```
Column {
    Text(status)

    Button(onClick = {
        scope.launch {
            status = "Yuklanmoqda..."
            downloadData() // 2 soniya kutadi
            status = "Ma'lumot yuklandi!"
        }
    }) {
        Text("Yuklash")
    }
}
```

```
}
```

```
suspend fun downloadData() {
delay(2000)
println("Ma'lumot yuklandi")
}
```

Natija:

1. Tugma bosiladi → "Yuklanmoqda..." ko'rsatiladi
2. 2 soniya kutiladi (lekin UI ishlaydi!)
3. "Ma'lumot yuklandi!" ko'rsatiladi

--- YANA BIR NECHA SUSPEND MISOLLARI ---

API dan ma'lumot olish:

```
suspend fun getUserData(userId: Int): User {
delay(1000) // Network kutish taqlidi
return User(id = userId, name = "Ali")
}
```

```
// Ishlatish:
coroutineScope.launch {
val user = getUserData(123)
println(user.name) // "Ali"
}
```

Ketma-ket operatsiyalar:

```
suspend fun loginProcess(email: String, password: String) {
// 1. Login qilish
delay(1000)
val token = "abc123"
```

```
// 2. Profil olish
delay(1000)
val profile = getProfile(token)

// 3. Ma'lumotlarni saqlash
delay(500)
saveToDatabase(profile)
```

```
}
```

Parallel operatsiyalar:

```
suspend fun loadDashboard() {
coroutineScope {
// Barchasi bir vaqtda boshlanadi
val user = async { loadUser() }      // 1 soniya
val posts = async { loadPosts() }    // 2 soniya
val stats = async { loadStats() }    // 1.5 soniya
```

```
    // Barchasi tugashini kutadi
    showDashboard(user.await(), posts.await(), stats.await())
    // Jami vaqt: 2 soniya (eng uzoq operatsiya)
}
```

```
}
```

--- QOIDALAR ---

✅Suspend funksiyani chaqirish mumkin:

```
// 1. Boshqa suspend funksiya ichida
suspend fun function1() {
function2() // OK
}

suspend fun function2() {
delay(1000)
}

// 2. Coroutine ichida
fun normalFunction() {
coroutineScope.launch {
function2() // OK
}
}
```

❌Suspend funksiyani chaqirib bo'lmaydi:

```
fun normalFunction() {
function2() // XATO! Coroutine kerak
}

suspend fun function2() {
delay(1000)
}
```

--- REAL ANDROID MISOL ---

```
class ProductRepository {
```

```
// Suspend funksiyalar
suspend fun getProducts(): List<Product> {
    delay(2000) // API so'rovi taqlidi
    return listOf(
        Product(1, "Telefon"),
        Product(2, "Noutbuk")
    )
}

suspend fun saveProduct(product: Product) {
    delay(1000) // Database ga saqlash taqlidi
    println("${product.name} saqlandi")
}
```

}

// ViewModel'da ishlatish
class ProductViewModel : ViewModel() {
private val repository = ProductRepository()

```
fun loadProducts() {
    viewModelScope.launch {
        val products = repository.getProducts() // Suspend funksiya
        _productsState.value = products
    }
}

fun addProduct(product: Product) {
    viewModelScope.launch {
        repository.saveProduct(product) // Suspend funksiya
        loadProducts() // Yangilash
    }
}
```

}

## ==================================================
## 4-BO'LIM: suspend fun VS launch VS async/await

Bu 3 ta turli narsa!

--- QISQACHA FARQ ---

| Nima? | Vazifasi | Qachon ishlatiladi |
|---|---|---|
| suspend fun | Funksiya turi | Kutish mumkin bo'lgan funksiya yaratish |
| launch | Coroutine boshlash | Natija kerak emas, faqat ish bajarish |
| async/await | Coroutine boshlash | Natija kerak bo'lganda |

--- 1. SUSPEND FUN - Bu Funksiya Turi ---

```
// Bu oddiy funksiya EMAS, bu SUSPEND funksiya
suspend fun downloadData(): String {
delay(2000)
return "Ma'lumot"
}
```

// Bu yerda hech qanday coroutine yo'q!
// Bu shunchaki funksiya deklaratsiyasi

Xususiyatlari:
❌O'zi coroutine yaratmaydi
❌O'zi ishga tushmaydi
✅Faqat boshqa birov uni chaqirishi kerak

--- 2. LAUNCH - Coroutine Boshlash (Natijasiz) ---

```
fun myFunction() {
coroutineScope.launch {
// Bu yerda coroutine BOSHLANDI
val data = downloadData() // suspend funksiyani chaqirish
println(data)
}
// Bu kod darhol ishga tushadi (kutmaydi)
println("Launch qilindi")
}
```

Xususiyatlari:
✅Coroutine yaratadi va ishga tushiradi
✅Natija qaytarmaydi (return yo'q)
✅"Fire and forget" (yubordi va unutdi)
✅Parallel ishlaydi

--- 3. ASYNC/AWAIT - Coroutine Boshlash (Natija bilan) ---

fun myFunction() {
coroutineScope.launch {
val result = async {
// Bu yerda coroutine BOSHLANDI
downloadData() // suspend funksiyani chaqirish
}

```
    val data = result.await() // Natijani kutish
    println(data)
}
```

}

Xususiyatlari:
✅Coroutine yaratadi va ishga tushiradi
✅Natija qaytaradi (return bor)
✅await() bilan natijani olish kerak
✅Parallel ishlaydi

--- REAL MISOLDA FARQLAR ---

Misol 1: Oddiy vazifa (launch)

```
Button(onClick = {
scope.launch {
// Faqat xabar ko'rsatish, natija kerak emas
showSnackbar("Salom")
delay(2000)
println("Tugadi")
}
// Bu kod darhol ishga tushadi
println("Tugma bosildi")
})
```

```
// Natija:
// "Tugma bosildi" (darhol)
// "Tugadi" (2 soniyadan keyin)
```

Misol 2: Natija kerak (async/await)

```
Button(onClick = {
scope.launch {
// Natija kerak bo'lganda async
val result = async {
calculateSum(10, 20)
}
```

```
    val sum = result.await() // Natijani olish
    println("Natija: $sum") // 30
}
```

```
})
```

```
suspend fun calculateSum(a: Int, b: Int): Int {
delay(1000)
return a + b
}
```

--- ULARNI BIRGA ISHLATISH ---

```
// 1. Suspend funksiya yaratish
suspend fun getUser(id: Int): User {
delay(1000)
return User(id, "Ali")
}
```

```
suspend fun getPosts(userId: Int): List<Post> { delay(1500) return listOf(Post(1, "Post 1")) }
```

```
// 2. Launch bilan chaqirish (ketma-ket)
fun loadDataSequential() {
scope.launch {
val user = getUser(1)      // 1 soniya kutadi
val posts = getPosts(1)    // 1.5 soniya kutadi
// Jami: 2.5 soniya
showProfile(user, posts)
}
}
```

```
// 3. Async bilan chaqirish (parallel)
fun loadDataParallel() {
scope.launch {
val userDeferred = async { getUser(1) }     // Boshlanadi
val postsDeferred = async { getPosts(1) }   // Boshlanadi
```

```
    val user = userDeferred.await()   // Kutadi
    val posts = postsDeferred.await() // Kutadi
    // Jami: 1.5 soniya (eng uzoq operatsiya)
    showProfile(user, posts)
}
```

```
}
```

--- KETMA-KET VS PARALLEL ---

Ketma-ket (launch ichida oddiy chaqirish):

```
scope.launch {
// Har biri navbat bilan
val step1 = downloadFile1() // 2 soniya
val step2 = downloadFile2() // 3 soniya
val step3 = downloadFile3() // 1 soniya
// Jami: 6 soniya
}
```

Parallel (async/await):

```
scope.launch {
// Hamsi bir vaqtda
val file1 = async { downloadFile1() } // 2 soniya
val file2 = async { downloadFile2() } // 3 soniya
val file3 = async { downloadFile3() } // 1 soniya
```

```
val f1 = file1.await()
val f2 = file2.await()
val f3 = file3.await()
// Jami: 3 soniya (eng uzoq operatsiya)
```

```
}
```

--- REAL ANDROID MISOL ---

```
class UserViewModel : ViewModel() {
```

```
// suspend funksiya - faqat deklaratsiya
suspend fun loginUser(email: String, password: String): User {
    delay(2000) // API so'rovi
    return User(1, "Ali")
}

// launch - natija kerak emas
fun sendAnalytics(event: String) {
    viewModelScope.launch {
        delay(500)
        analyticsService.send(event)
        // Natija kerak emas, shunchaki yuborildi
    }
}

// launch + suspend - oddiy ishlash
fun login(email: String, password: String) {
    viewModelScope.launch {
        try {
            _isLoading.value = true

            val user = loginUser(email, password) // suspend chaqirish
            _userState.value = user

        } catch (e: Exception) {
            _error.value = e.message
        } finally {
            _isLoading.value = false
        }
    }
}

// async - bir necha natija kerak
fun loadDashboard() {
    viewModelScope.launch {
        try {
            _isLoading.value = true

            // Parallel yuklash
            val profile = async { getProfile() }
            val posts = async { getPosts() }
            val notifications = async { getNotifications() }

            // Natijalarni olish
            _profileState.value = profile.await()
            _postsState.value = posts.await()
            _notificationsState.value = notifications.await()

        } catch (e: Exception) {
            _error.value = e.message
        } finally {
            _isLoading.value = false
        }
    }
}
```
}

--- QACHON NIMANI ISHLATISH ---

suspend fun ishlatish:

// ✅Kutish kerak bo'lgan har qanday funksiya
suspend fun fetchData(): Data { ... }
suspend fun saveToDatabase(data: Data) { ... }
suspend fun processImage(image: Bitmap): Bitmap { ... }

launch ishlatish:

// ✅Natija kerak emas
launch { sendLog("User logged in") }
launch { updateCache() }
launch { showNotification() }

async/await ishlatish:

// ✅Natija kerak va parallel ishlash kerak
val user = async { getUser() }
val posts = async { getPosts() }
showProfile(user.await(), posts.await())

--- XATO MISOLLAR ---

❌XATO 1: suspend funksiyani oddiy chaqirish

```
fun myFunction() {
val data = downloadData() // XATO! Coroutine kerak
}
```

❌XATO 2: launch'dan natija olishga harakat

```
val result = launch {
downloadData()
} // result - Job, Data emas!
```

✅TO'G'RI:

```
// 1. suspend funksiyani coroutine ichida
fun myFunction() {
scope.launch {
val data = downloadData() // TO'G'RI
}
}

// 2. Natija olish uchun async
fun myFunction() {
scope.launch {
val result = async {
downloadData()
}
val data = result.await() // TO'G'RI
}
}
```

**================================================================**

## XULOSA

ODDIY TILDA:

- suspend fun = Vaqt talab qiladigan funksiya (mashina) 🚗
- launch = Mashinani ishga tushirish, qayerga borishi muhim emas 🚀
- async/await = Mashinani ishga tushirish va qaytib kelishini kutish 🎯

QOIDA:

1. Avval suspend fun yarat (funksiya)
2. Uni launch yoki async bilan ishga tushir (coroutine)
3. Natija kerak bo'lsa async/await, kerak bo'lmasa launch ishlat

COROUTINE = Bir vaqtning o'zida bir necha ishni qilish imkoniyati!

delay() - thread'ni bloklamaydi ✅
Thread.sleep() - thread'ni bloklaydi ❌
suspend funksiyani faqat coroutine yoki boshqa suspend funksiya ichida chaqirish mumkin
Android'da network, database, fayl operatsiyalari uchun ishlatiladi

suspend = "Men vaqt talab qiladigan ishman, meni coroutine ichida chaqiring!"

## ================================================================
## MUHIM ESLATMALAR

1. Coroutine - bu yengil thread, ko'p xotira olmaydi
2. delay() - thread'ni bloklamaydi, Thread.sleep() bloklaydi
3. suspend fun - faqat coroutine ichida chaqiriladi
4. launch - natija qaytarmaydi, "fire and forget"
5. async/await - natija qaytaradi, parallel ishlash uchun
6. viewModelScope - ViewModel'da ishlatiladi
7. rememberCoroutineScope - Compose'da ishlatiladi
8. Dispatchers.Main - UI thread'da ishlash
9. Dispatchers.IO - Network, database, fayl operatsiyalari
10. Dispatchers.Default - Og'ir hisob-kitoblar

Coroutine ishlatilmaydigan joylarni topish qiyin!

Agar operatsiya 16ms dan ko'p vaqt olsa (1 frame), coroutine ishlatish kerak!

## ================================================================
## OXIRI