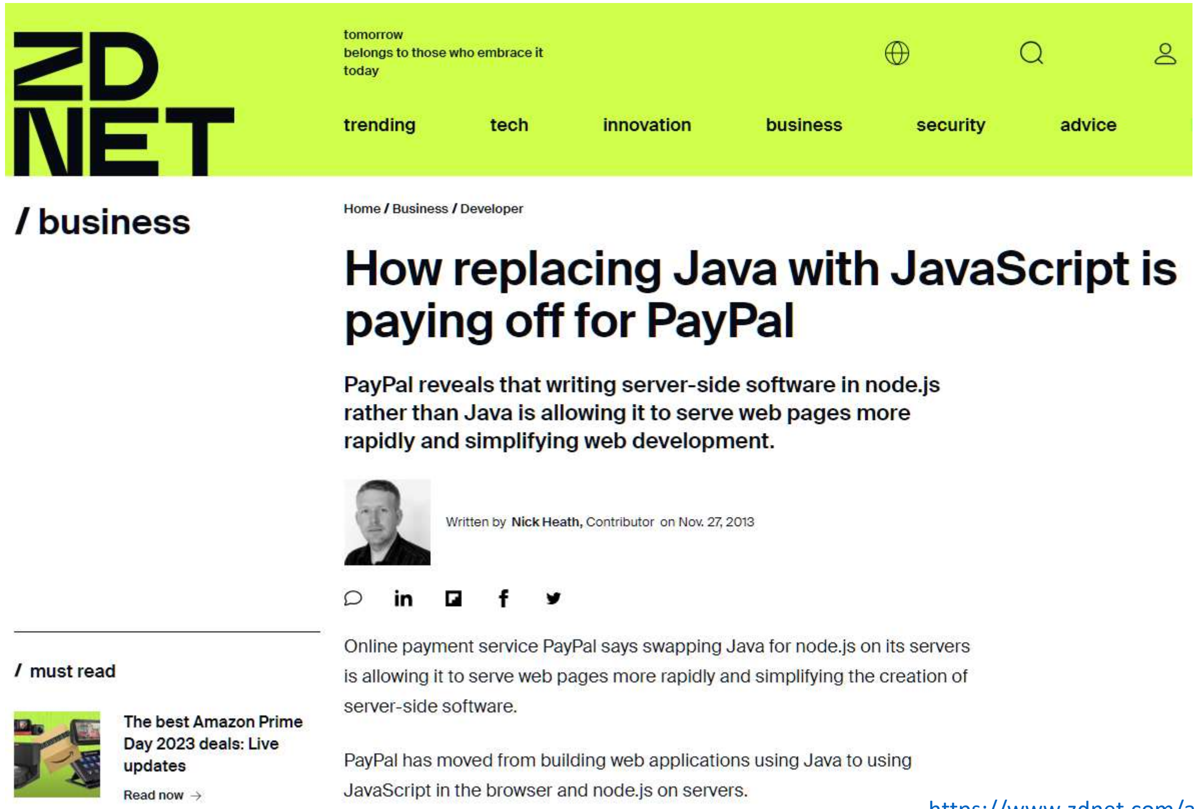
Async NIO

Paypal이 Java에서 Node.js로 간 이유 (2013.11.27 zdnet)



Home / Business / Developer

How replacing Java with JavaScript is paying off for PayPal

PayPal reveals that writing server-side software in node.js rather than Java is allowing it to serve web pages more rapidly and simplifying web development.

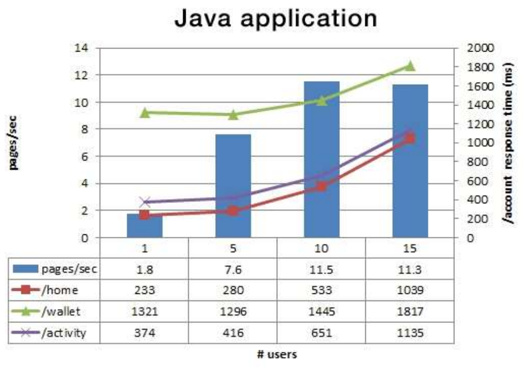
Written by **Nick Heath,** Contributor on Nov. 27, 2013

Online payment service PayPal says swapping Java for node.js on its servers is allowing it to serve web pages more rapidly and simplifying the creation of server-side software.

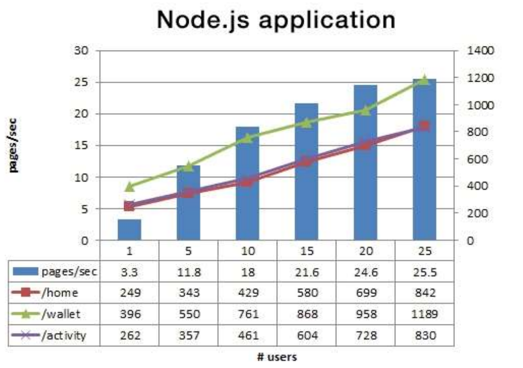
PayPal has moved from building web applications using Java to using JavaScript in the browser and node.js on servers.

The best Amazon Prime Day 2023 deals: Live updates

Read now →



	1	5	10	15
pages/sec	1.8	7.6	11.5	11.3
/home	233	280	533	1039
/wallet	1321	1296	1445	1817
/activity	374	416	651	1135



	1	5	10	15	20	25
pages/sec	3.3	11.8	18	21.6	24.6	25.5
/home	249	343	429	580	699	842
/wallet	396	550	761	868	958	1189
/activity	262	357	461	604	728	830

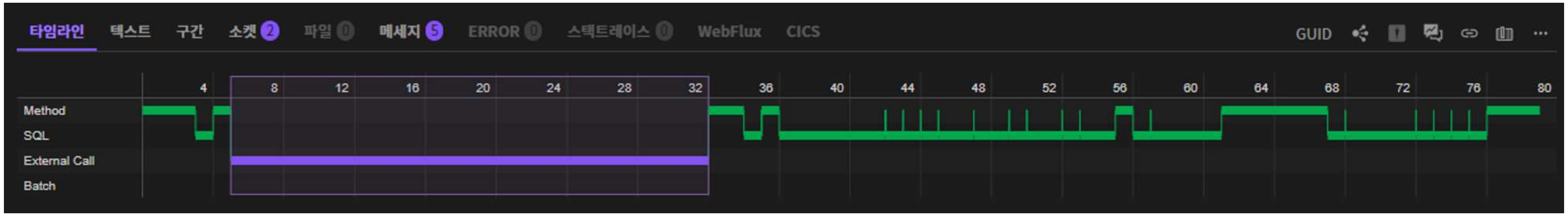
https://www.zdnet.com/article/how-replacing-java-with-javascript-is-paying-off-for-paypal

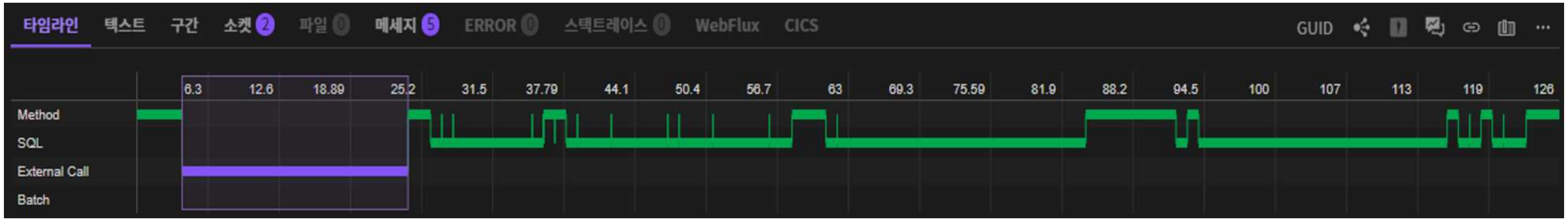
Async NIO

Non-blocking IO

Async Non-Blocking

Blocking request





Node.js

Non-blocking I/O

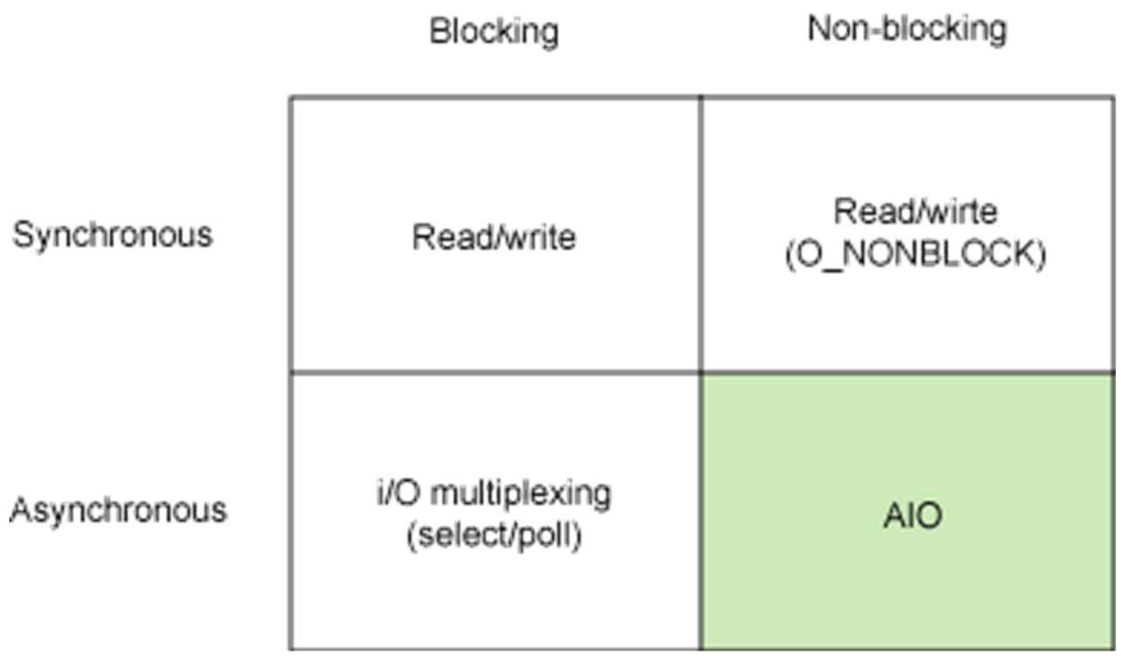
Asynchronous

Event loop

Async Non-Blocking

Boost application performance using asynchronous I/O

Learn when and how to use POSIX AIO API, 2006.8.28, M.Jones

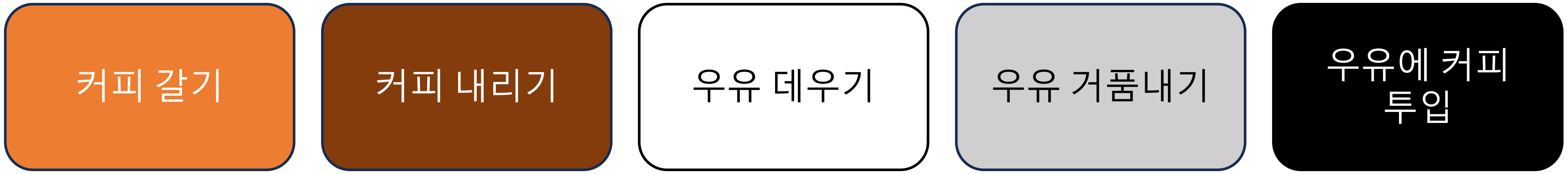


	Blocking	Non-blocking
Synchronous	Read/write	Read/wirte (O_NONBLOCK)
Asynchronous	i/O multiplexing (select/poll)	🟩 AIO

https://developer.ibm.com/articles/l-async

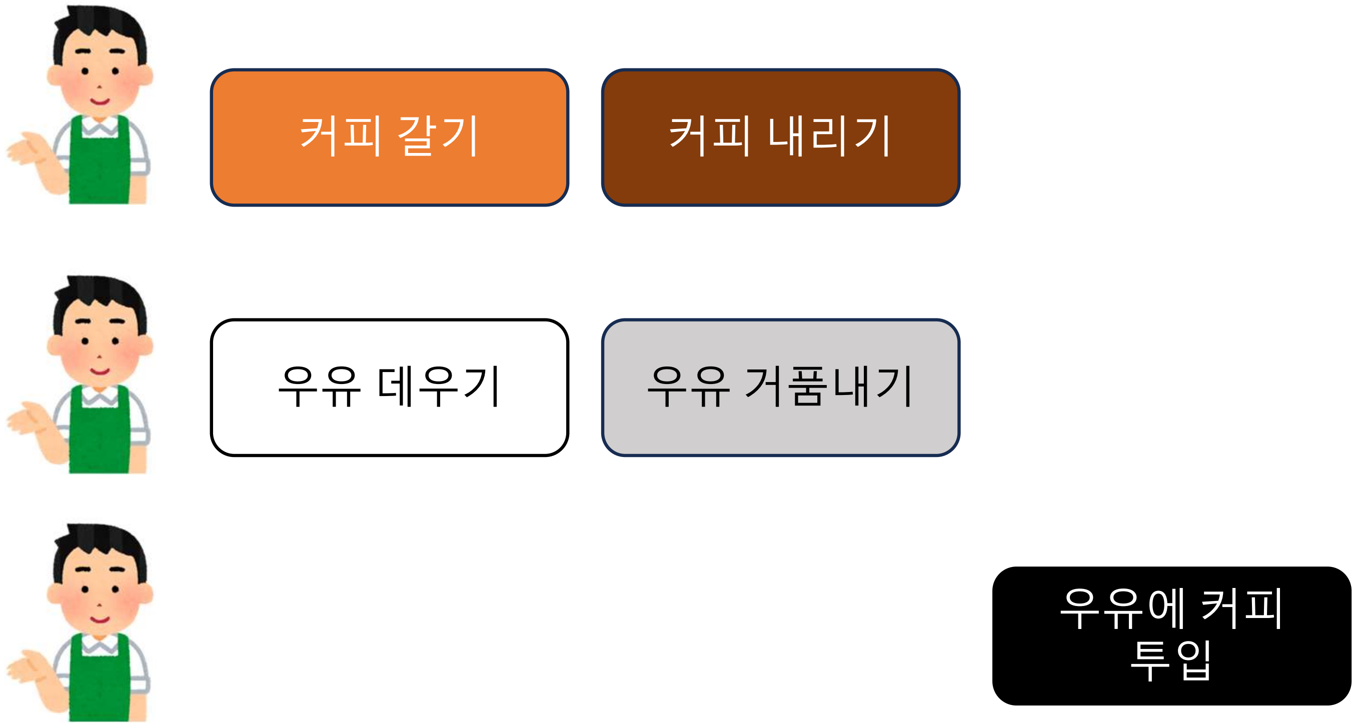
Async Non-Blocking

까페라떼 한 잔 주세요



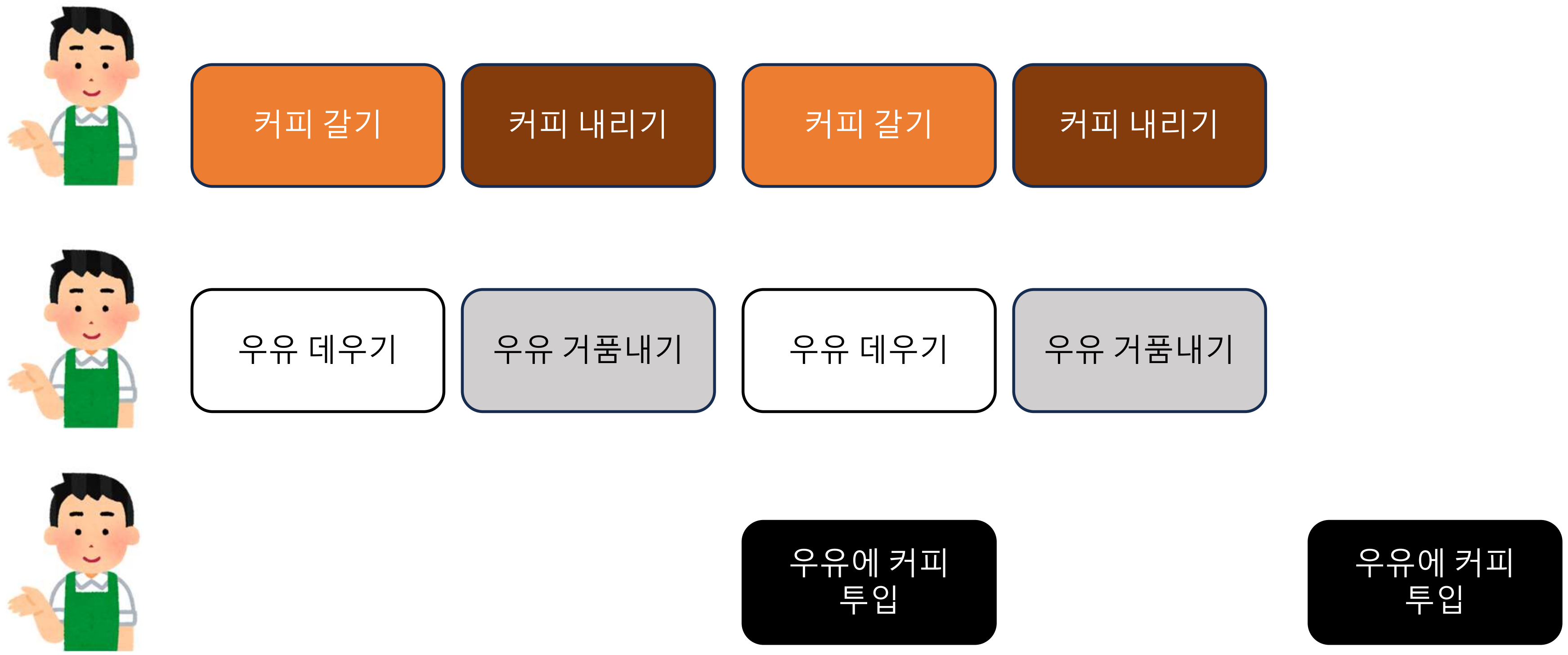
Async Non-Blocking

까페라떼 한 잔 주세요



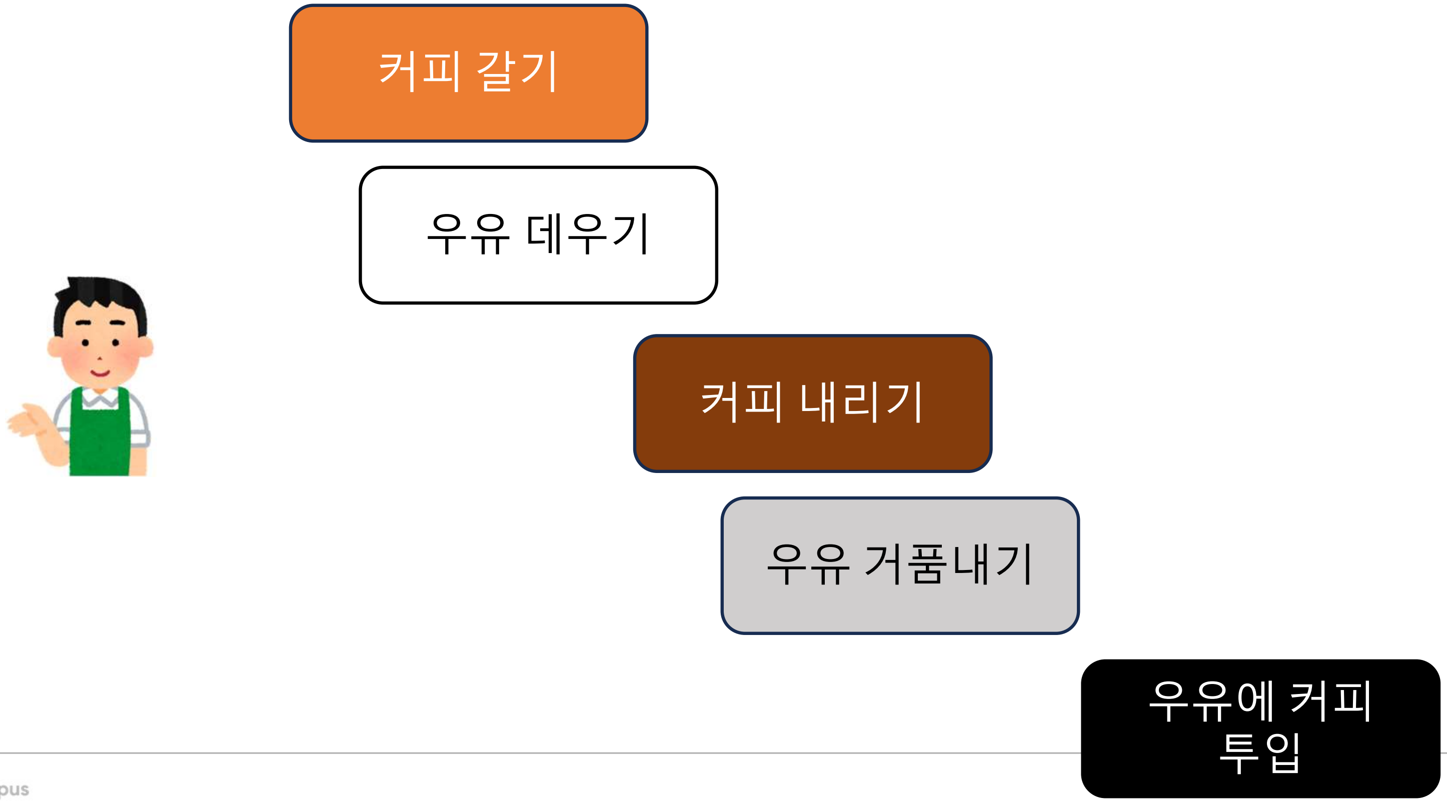
Async Non-Blocking

까페라떼 한 잔 주세요 **X 2**



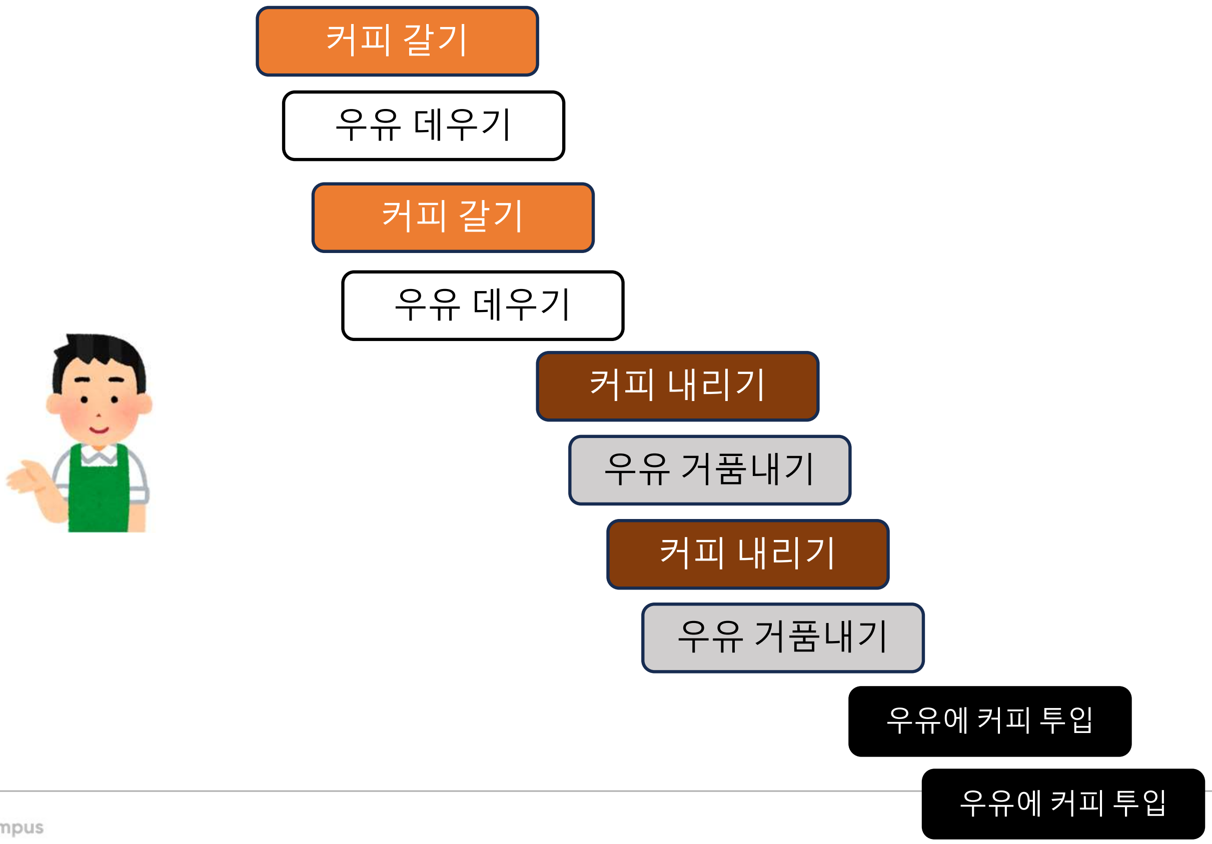
Async Non-Blocking

까페라떼 한 잔 주세요



Async Non-Blocking

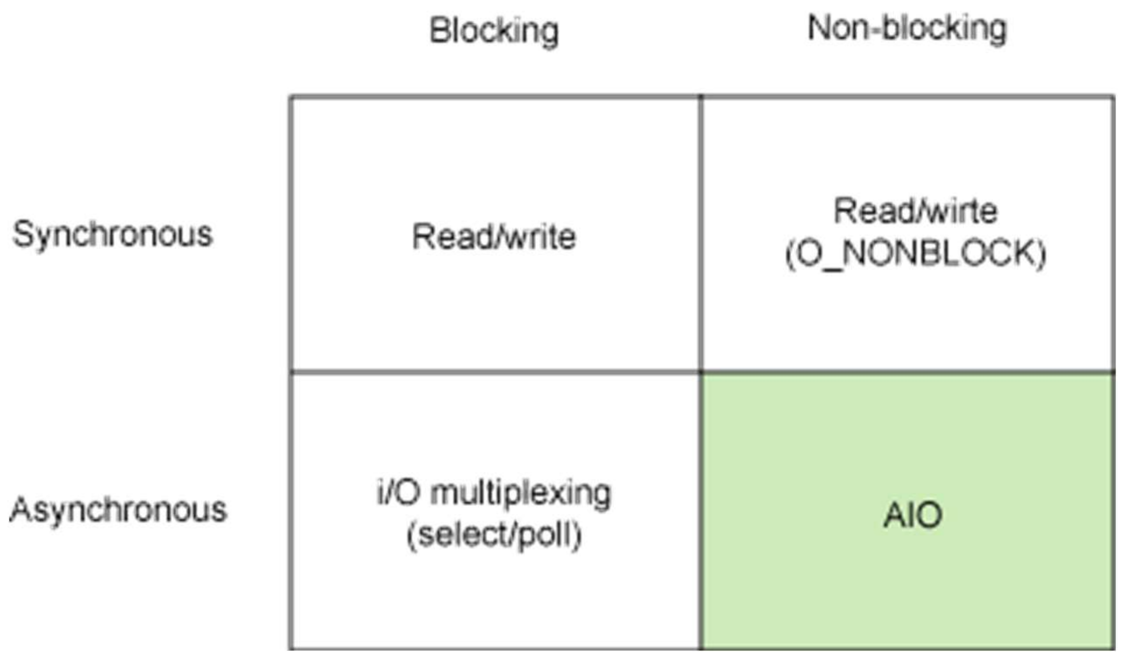
까페라떼 한 잔 주세요 **X 2**



Async Non-Blocking

Boost application performance using asynchronous I/O

Learn when and how to use POSIX AIO API, 2006.8.28, M.Jones

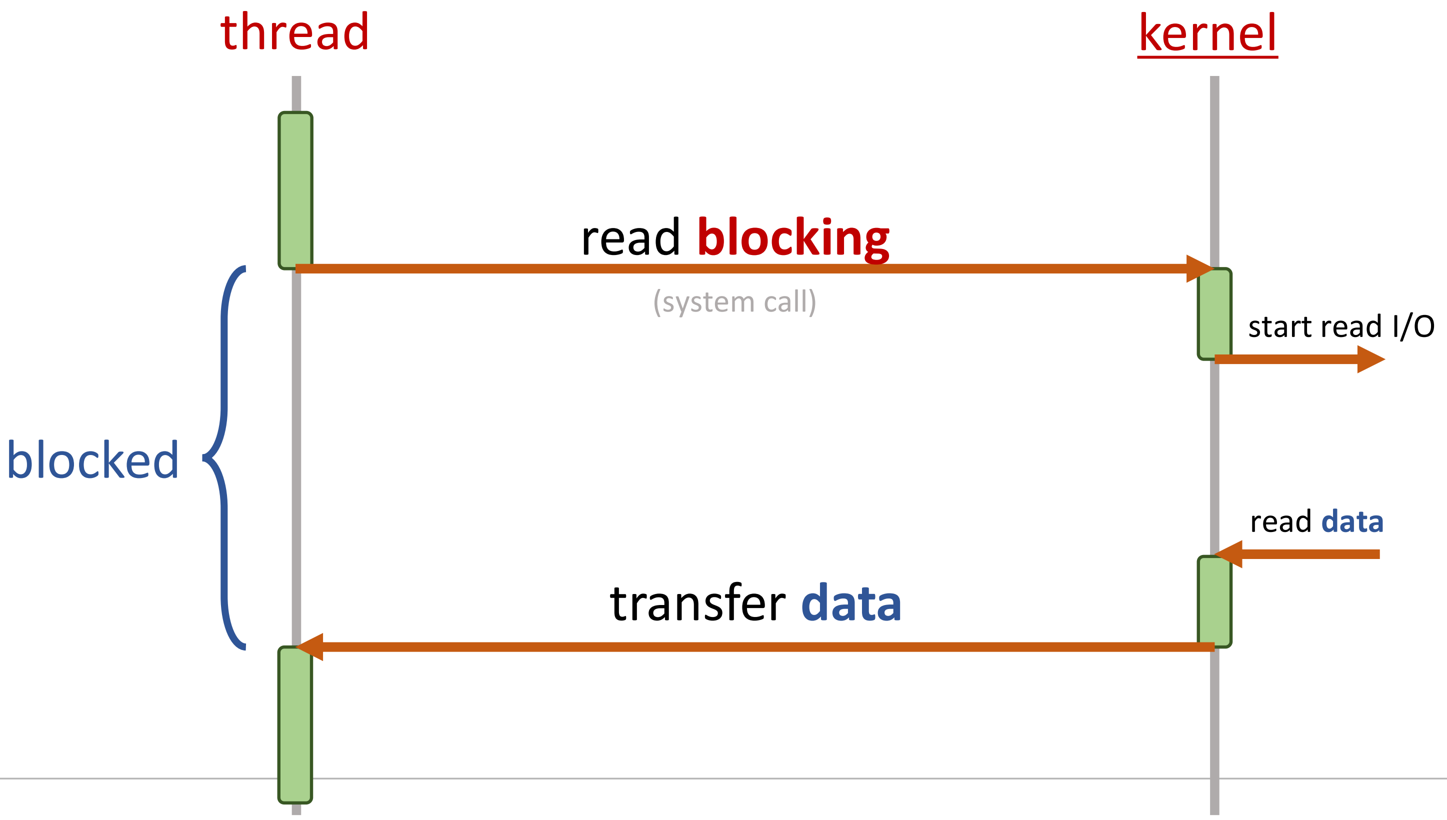


	Blocking	Non-blocking
Synchronous	Read/write	Read/wirte (O_NONBLOCK)
Asynchronous	i/O multiplexing (select/poll)	AIO 🟩

https://developer.ibm.com/articles/l-async

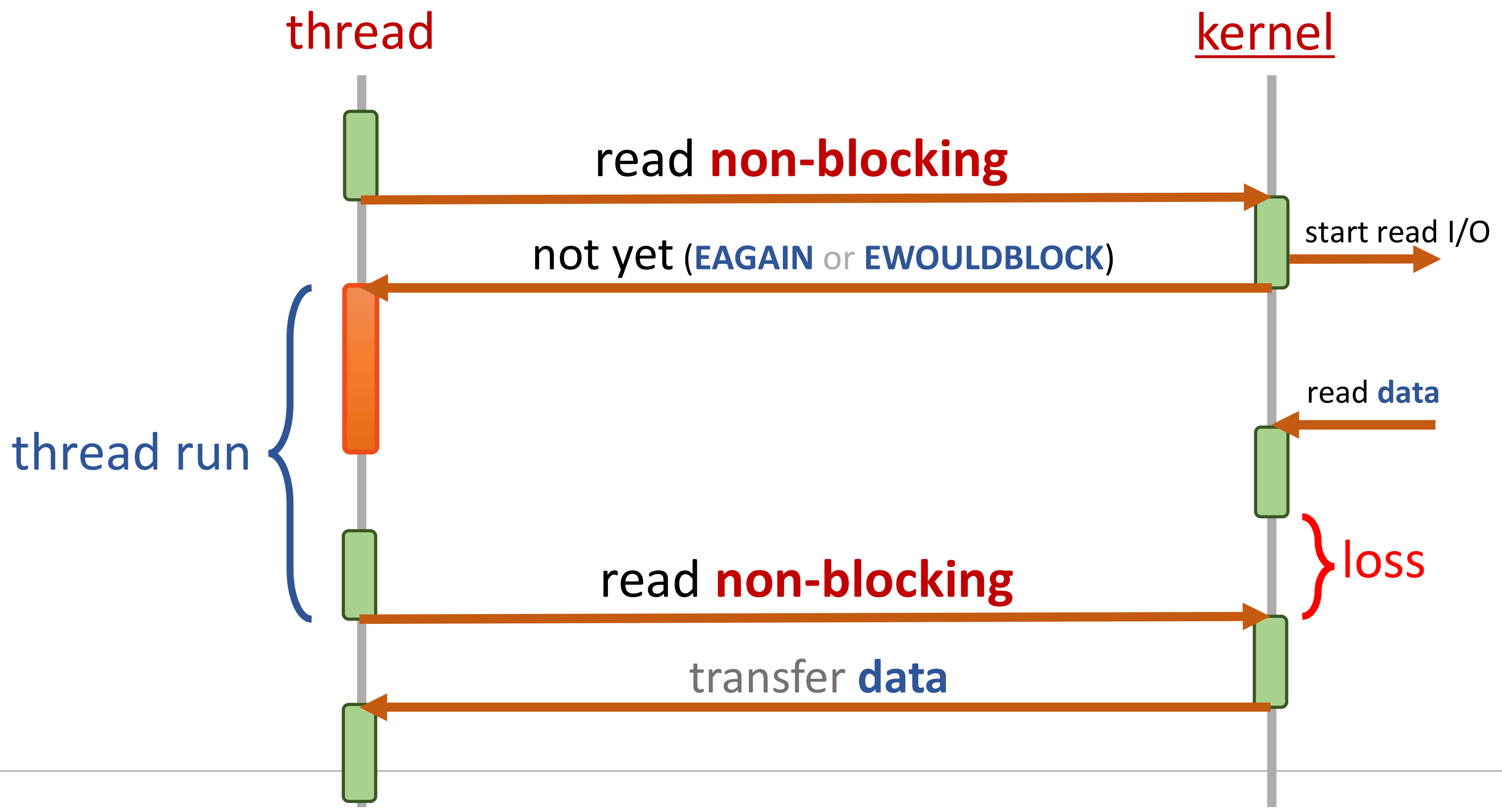
Async Non-Blocking

Sync / Blocking



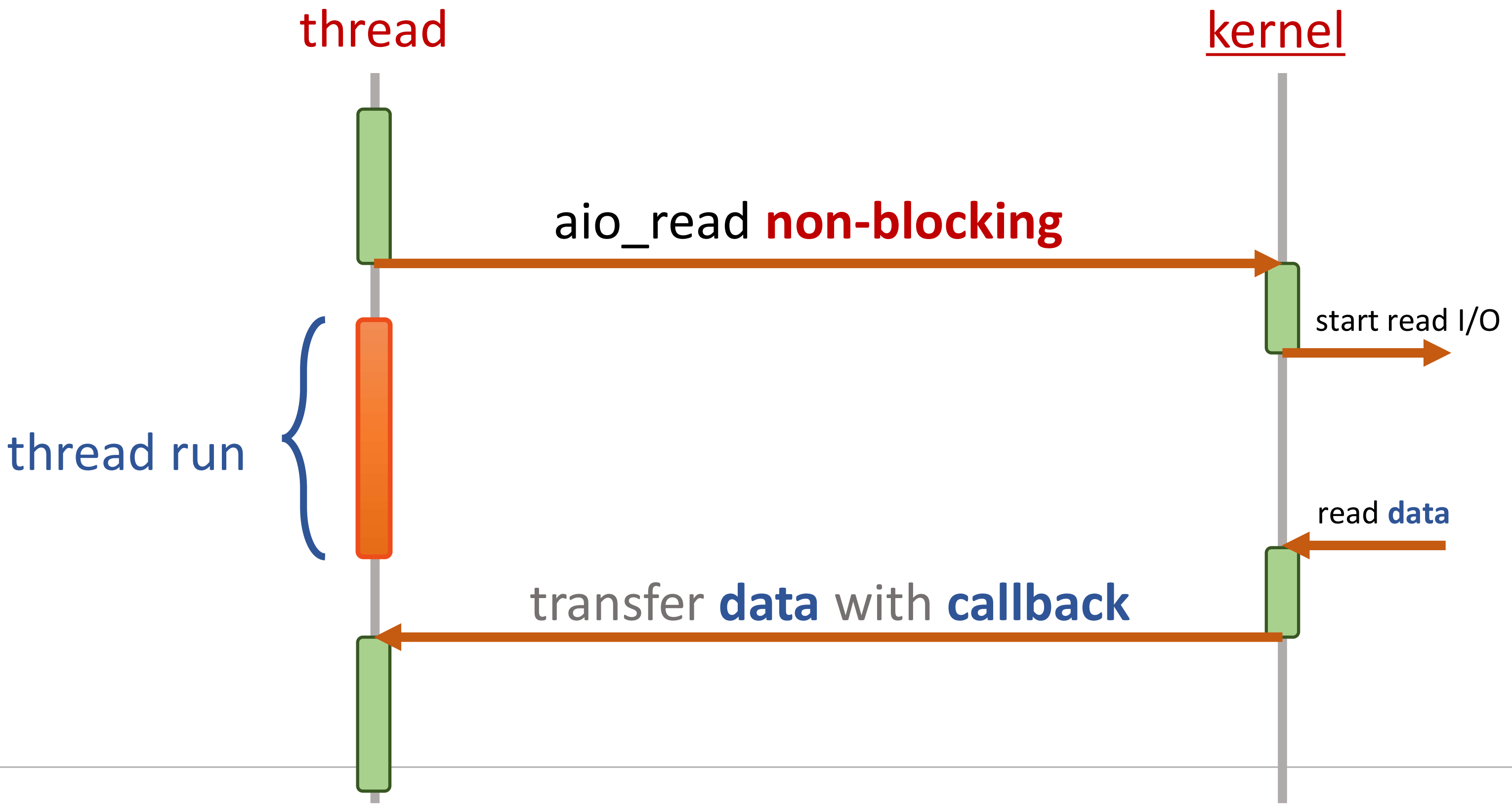
Async Non-Blocking

Sync / Non-Blocking



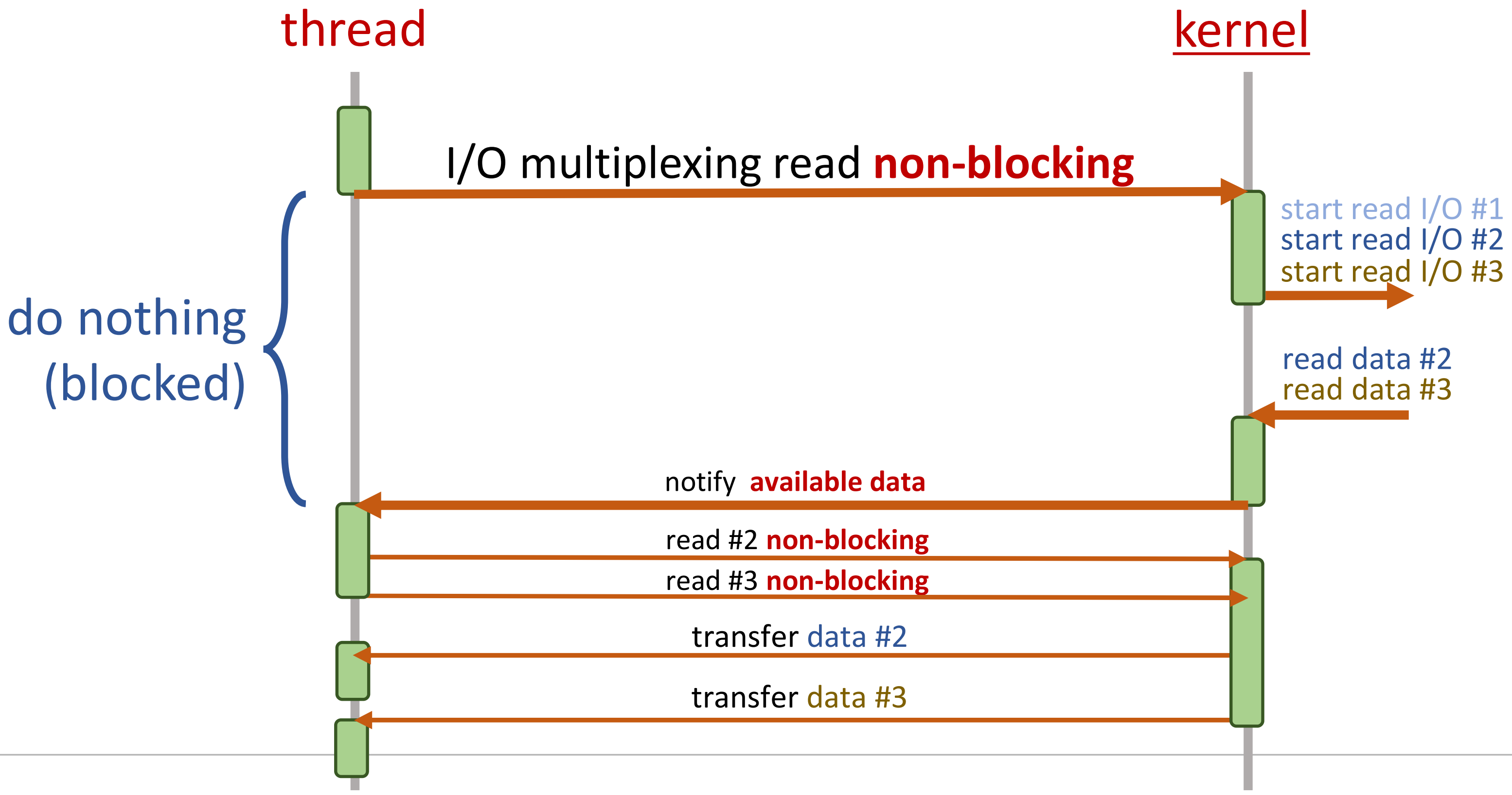
Async Non-Blocking

Async / Non-Blocking


thread
kernel
aio_read **non-blocking**
start read I/O
thread run
read **data**
transfer **data** with **callback**

Async Non-Blocking

Async / Blocking



구현 실습

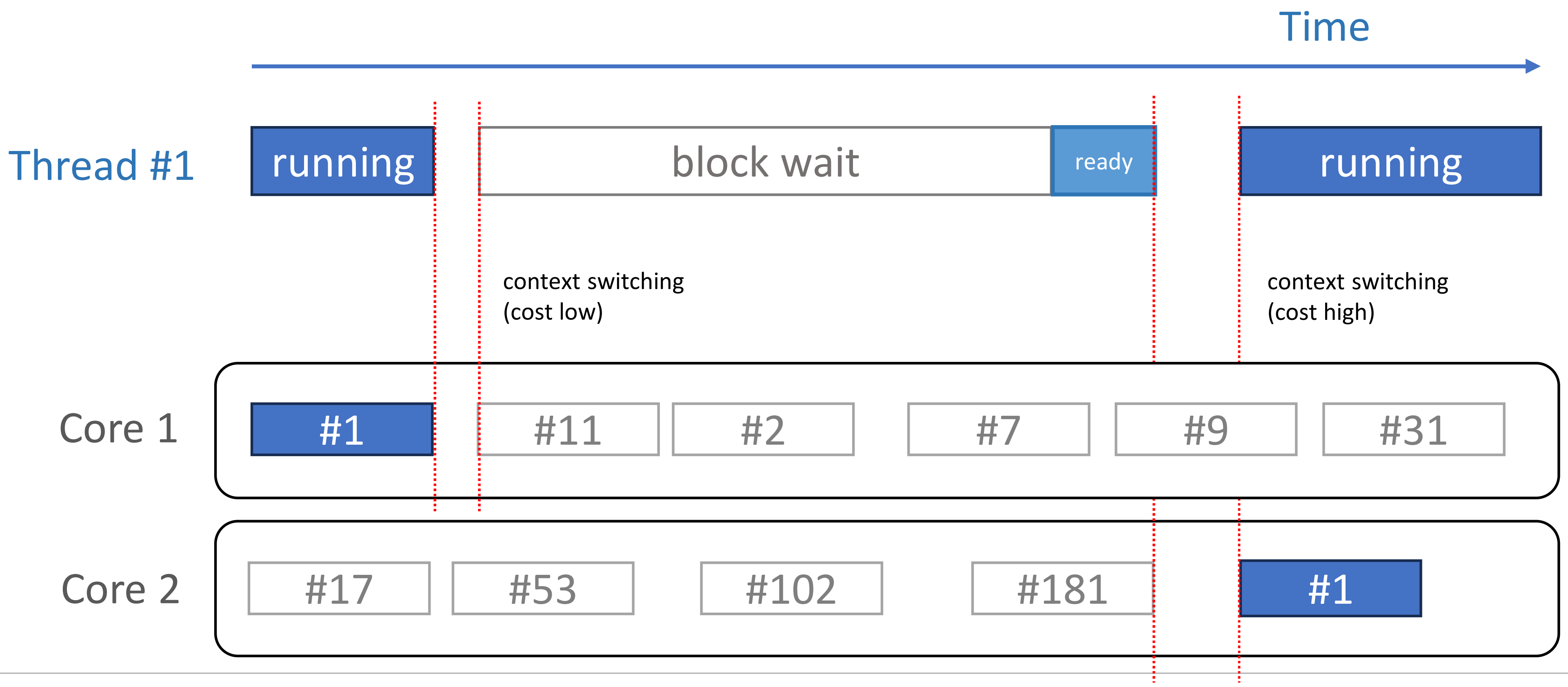
Coffee by Thread & Coroutine

Async NIO

Blocking IO in OS

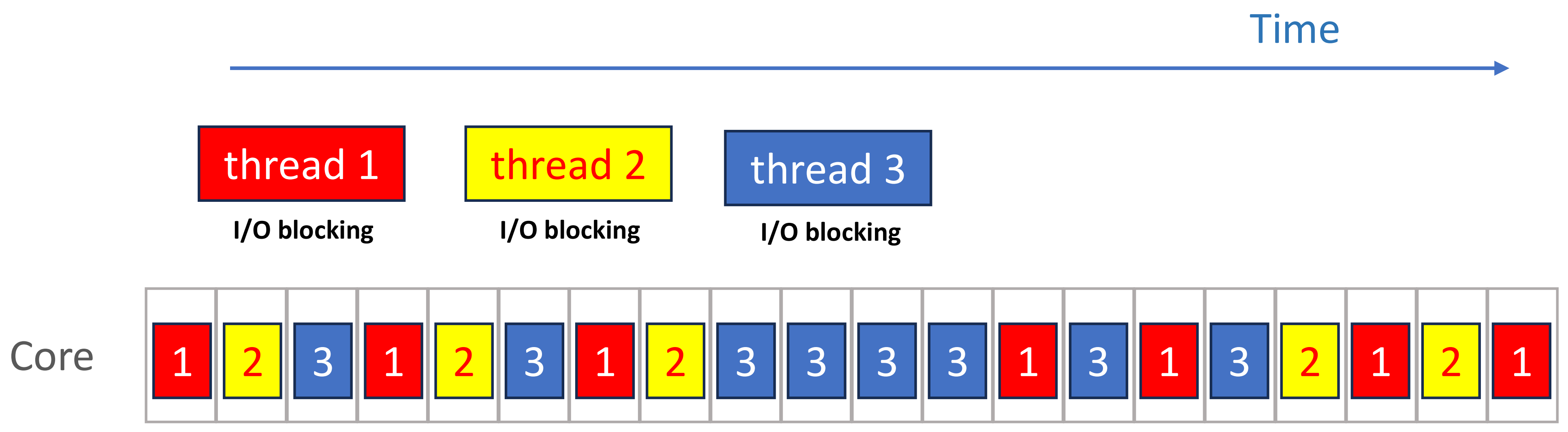
Blocking IO in OS

Synchronous request in OS



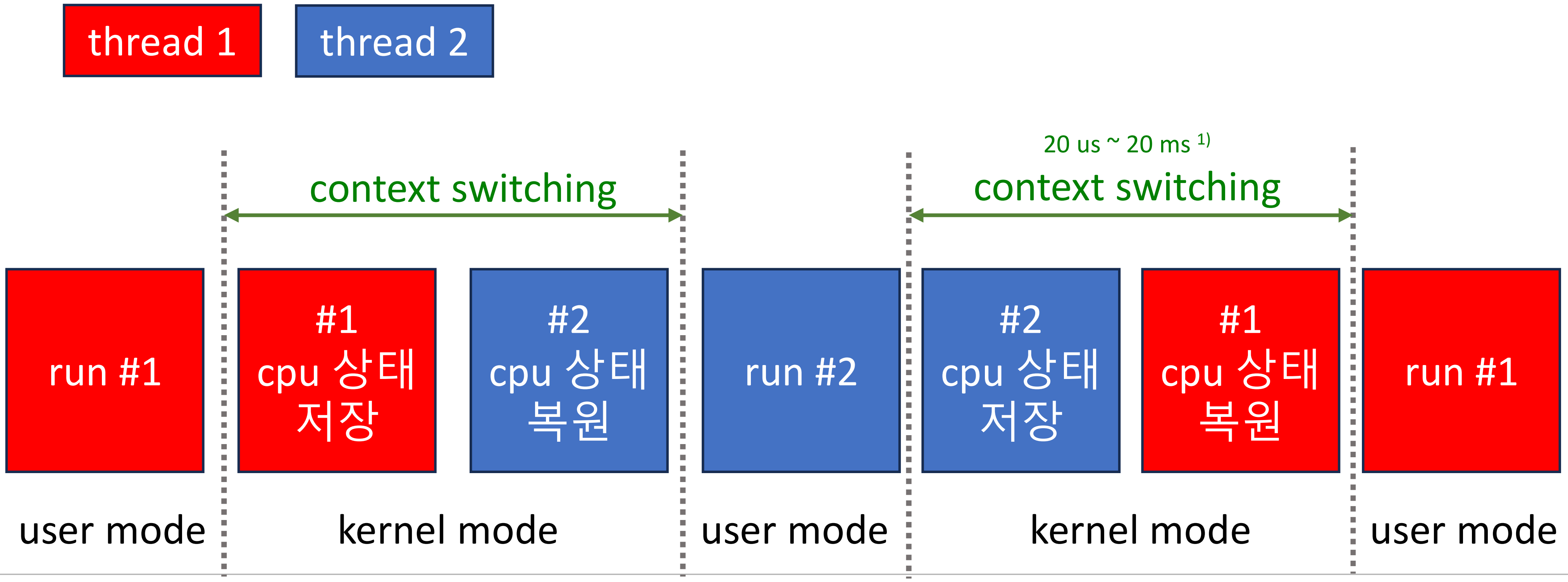
Blocking IO in OS

Time Slice



Blocking IO in OS

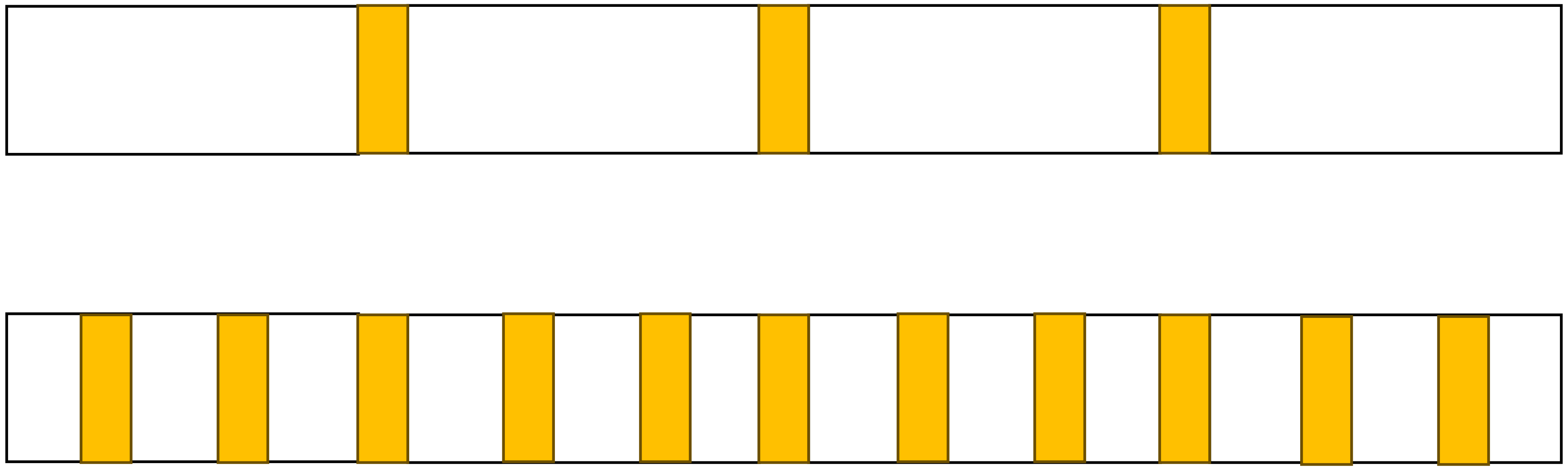
Thread context switching



1) https://blog.tsunanet.net/2010/11/how-long-does-it-take-to-make-context.html

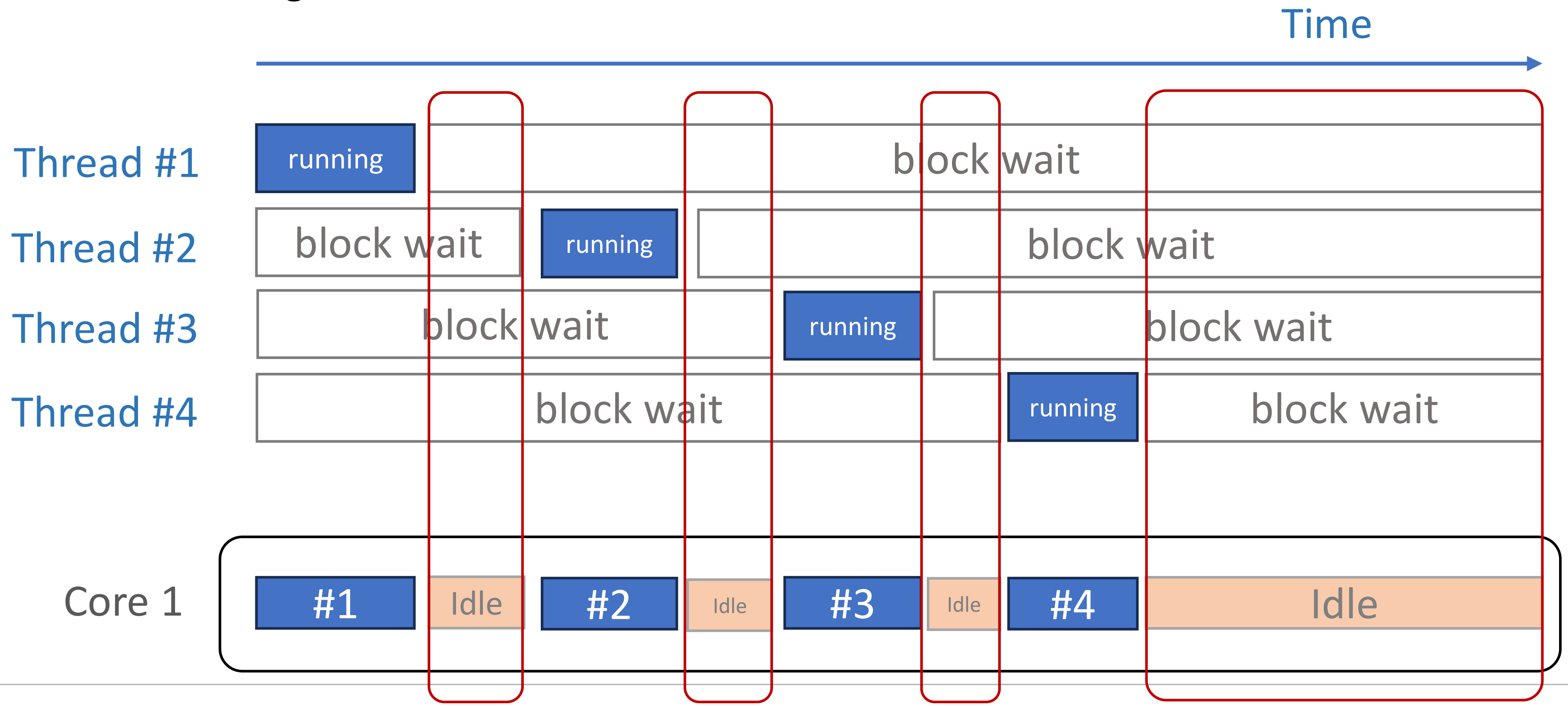
Blocking IO in OS

Thread context switching



Blocking IO in OS

Reduce working threads



Blocking IO in OS

Thread pool Dilemma

- Thread 를 늘리면
  - 메모리, CPU 부하로 성능 저하
- Thread 를 줄이면
  - 메모리, CPU는 충분하지만, thread 가 모자라서 처리율 저하

구현 실습

Add number

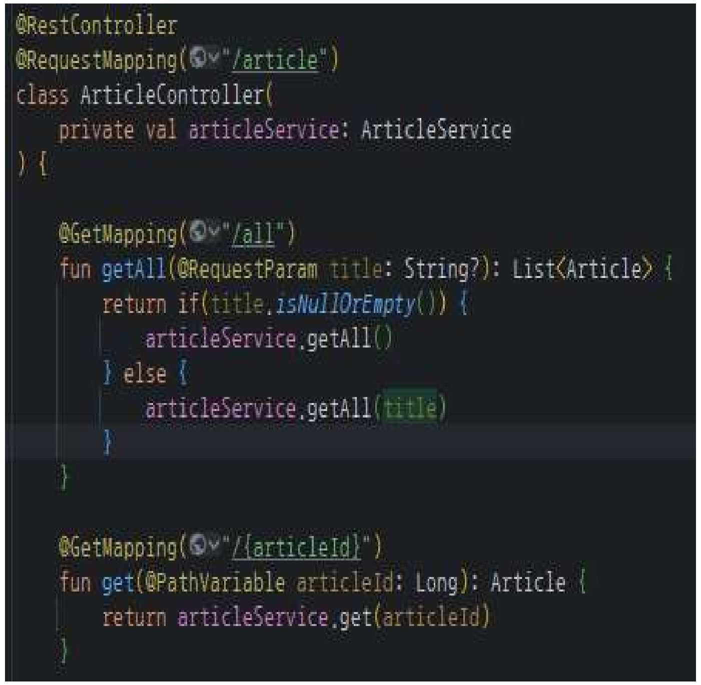
Thread & Coroutine

Async NIO

Coroutine

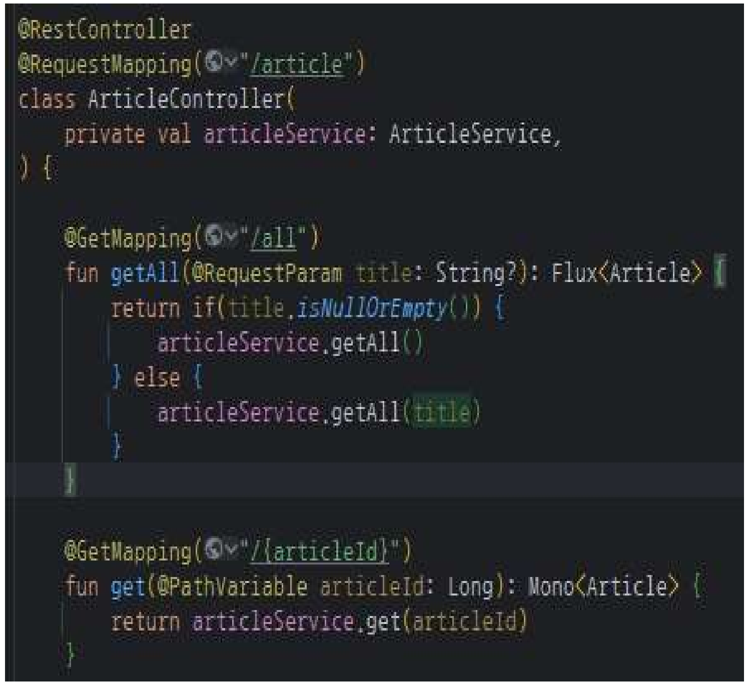
Controller

MVC



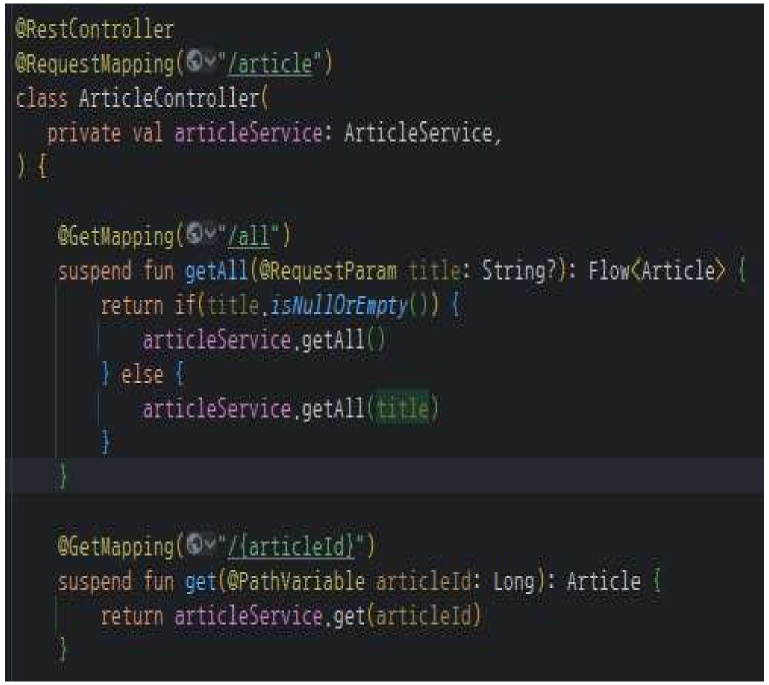
```
@RestController
@RequestMapping("/article")
class ArticleController(
    private val articleService: ArticleService
) {

    @GetMapping("/all")
    fun getAll(@RequestParam title: String?): List<Article> {
        return if(title.isNullOrEmpty()) {
            articleService.getAll()
        } else {
            articleService.getAll(title)
        }
    }

    @GetMapping("/{articleId}")
    fun get(@PathVariable articleId: Long): Article {
        return articleService.get(articleId)
    }
```

Reactor



```
@RestController
@RequestMapping("/article")
class ArticleController(
    private val articleService: ArticleService,
) {

    @GetMapping("/all")
    fun getAll(@RequestParam title: String?): Flux<Article> {
        return if(title.isNullOrEmpty()) {
            articleService.getAll()
        } else {
            articleService.getAll(title)
        }
    }

    @GetMapping("/{articleId}")
    fun get(@PathVariable articleId: Long): Mono<Article> {
        return articleService.get(articleId)
    }
```

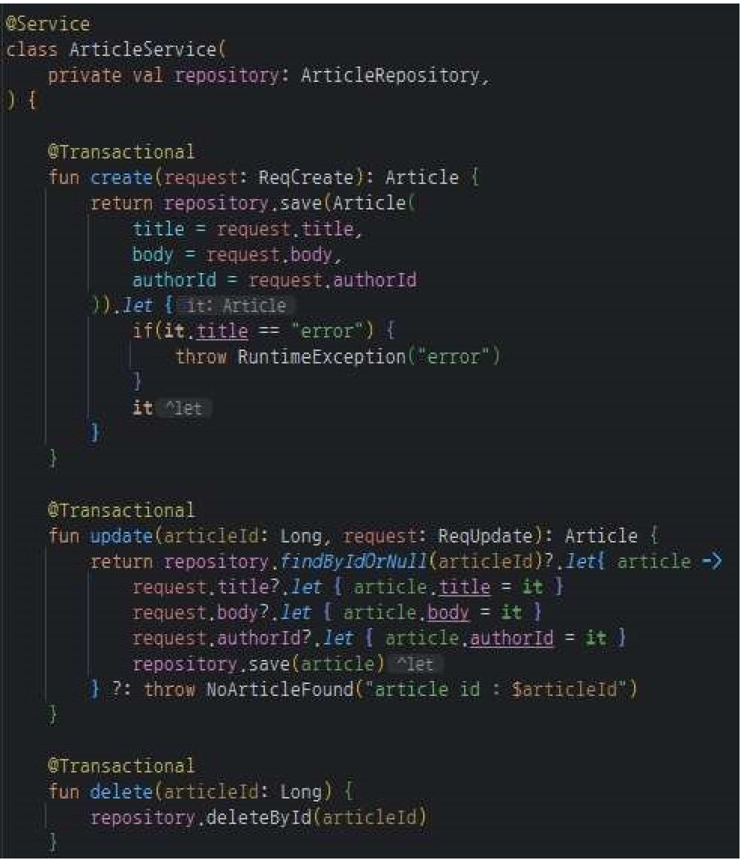
Coroutine



```
@RestController
@RequestMapping("/article")
class ArticleController(
    private val articleService: ArticleService,
) {

    @GetMapping("/all")
    suspend fun getAll(@RequestParam title: String?): Flow<Article> {
        return if(title.isNullOrEmpty()) {
            articleService.getAll()
        } else {
            articleService.getAll(title)
        }
    }

    @GetMapping("/{articleId}")
    suspend fun get(@PathVariable articleId: Long): Article {
        return articleService.get(articleId)
    }
```

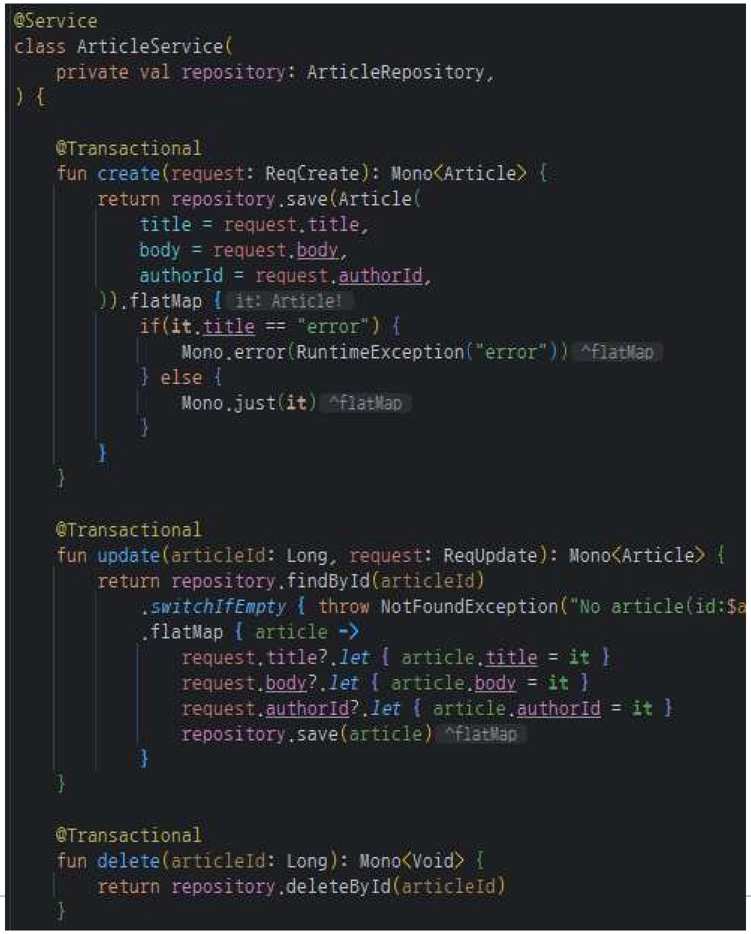
Service

MVC



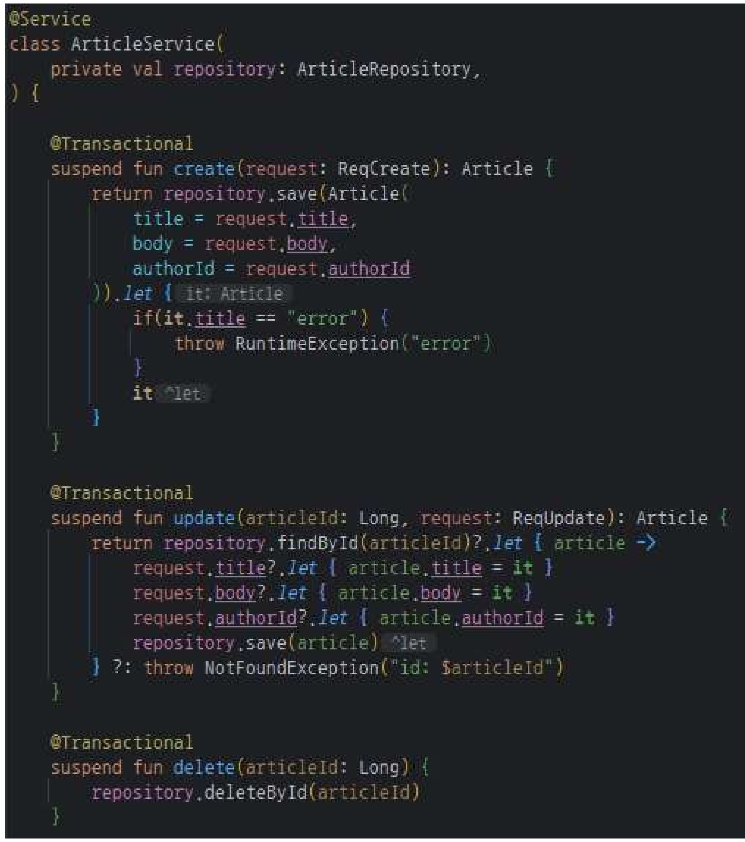
```kotlin
@Service
class ArticleService(
    private val repository: ArticleRepository,
) {

    @Transactional
    fun create(request: ReqCreate): Article {
        return repository.save(Article(
            title = request.title,
            body = request.body,
            authorId = request.authorId
        )).let { it: Article
            if(it.title == "error") {
                throw RuntimeException("error")
            }
            it ^let
        }
    }

    @Transactional
    fun update(articleId: Long, request: ReqUpdate): Article {
        return repository.findByIdOrNull(articleId)?.let{ article ->
            request.title?.let { article.title = it }
            request.body?.let { article.body = it }
            request.authorId?.let { article.authorId = it }
            repository.save(article) ^let
        } ?: throw NoArticleFound("article id : $articleId")
    }

    @Transactional
    fun delete(articleId: Long) {
        repository.deleteById(articleId)
    }
```

Reactor



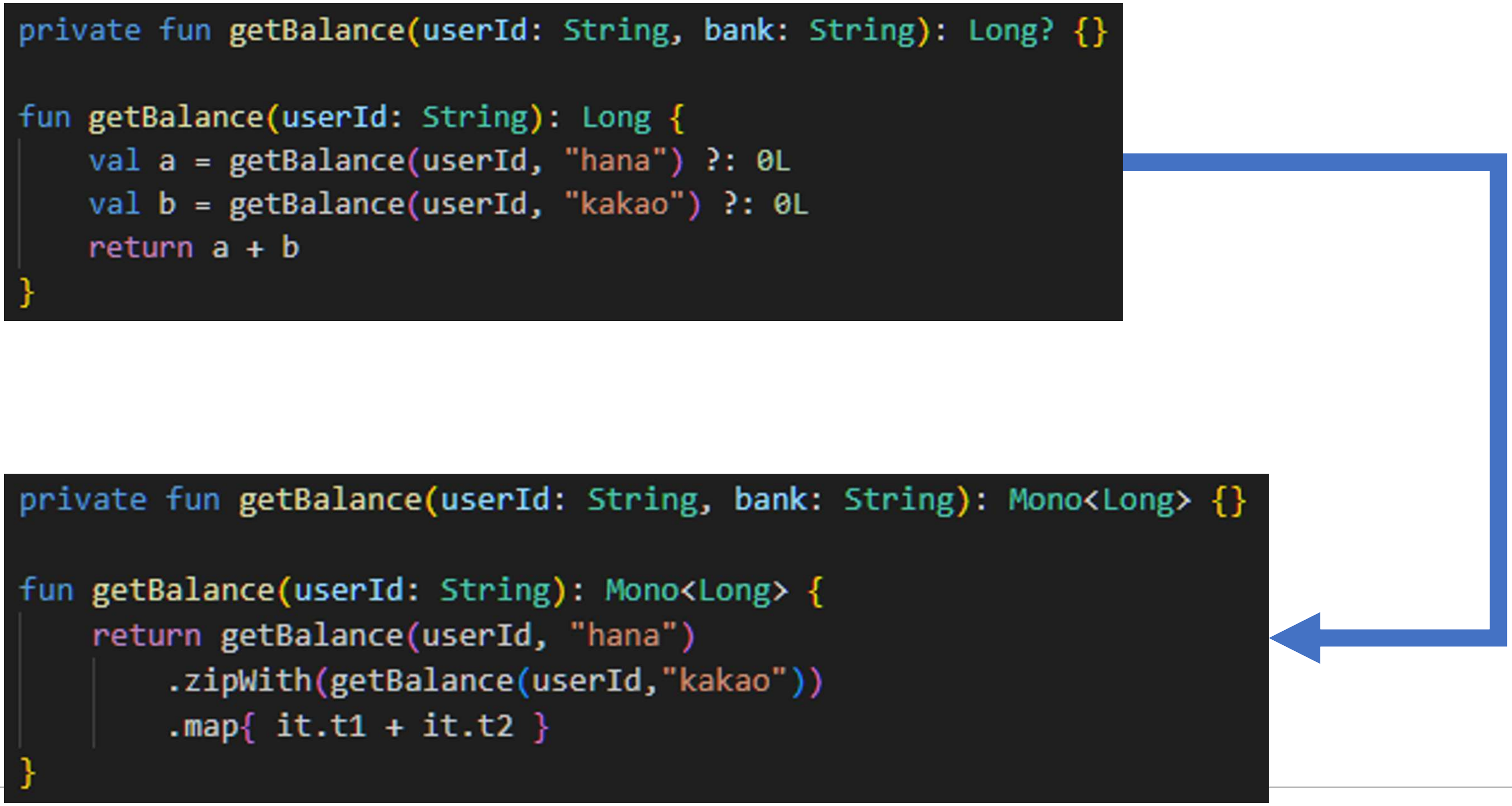
```kotlin
@Service
class ArticleService(
    private val repository: ArticleRepository,
) {

    @Transactional
    fun create(request: ReqCreate): Mono<Article> {
        return repository.save(Article(
            title = request.title,
            body = request.body,
            authorId = request.authorId,
        )).flatMap { it: Article!
            if(it.title == "error") {
                Mono.error(RuntimeException("error")) ^flatMap
            } else {
                Mono.just(it) ^flatMap
            }
        }
    }

    @Transactional
    fun update(articleId: Long, request: ReqUpdate): Mono<Article> {
        return repository.findById(articleId)
            .switchIfEmpty { throw NotFoundException("No article(id:$a
            .flatMap { article ->
                request.title?.let { article.title = it }
                request.body?.let { article.body = it }
                request.authorId?.let { article.authorId = it }
                repository.save(article) ^flatMap
            }
    }

    @Transactional
    fun delete(articleId: Long): Mono<Void> {
        return repository.deleteById(articleId)
    }
```

Coroutine



```kotlin
@Service
class ArticleService(
    private val repository: ArticleRepository,
) {

    @Transactional
    suspend fun create(request: ReqCreate): Article {
        return repository.save(Article(
            title = request.title,
            body = request.body,
            authorId = request.authorId
        )).let { it: Article
            if(it.title == "error") {
                throw RuntimeException("error")
            }
            it ^let
        }
    }

    @Transactional
    suspend fun update(articleId: Long, request: ReqUpdate): Article {
        return repository.findById(articleId)?.let { article ->
            request.title?.let { article.title = it }
            request.body?.let { article.body = it }
            request.authorId?.let { article.authorId = it }
            repository.save(article) ^let
        } ?: throw NotFoundException("id: $articleId")
    }

    @Transactional
    suspend fun delete(articleId: Long) {
        repository.deleteById(articleId)
    }
```

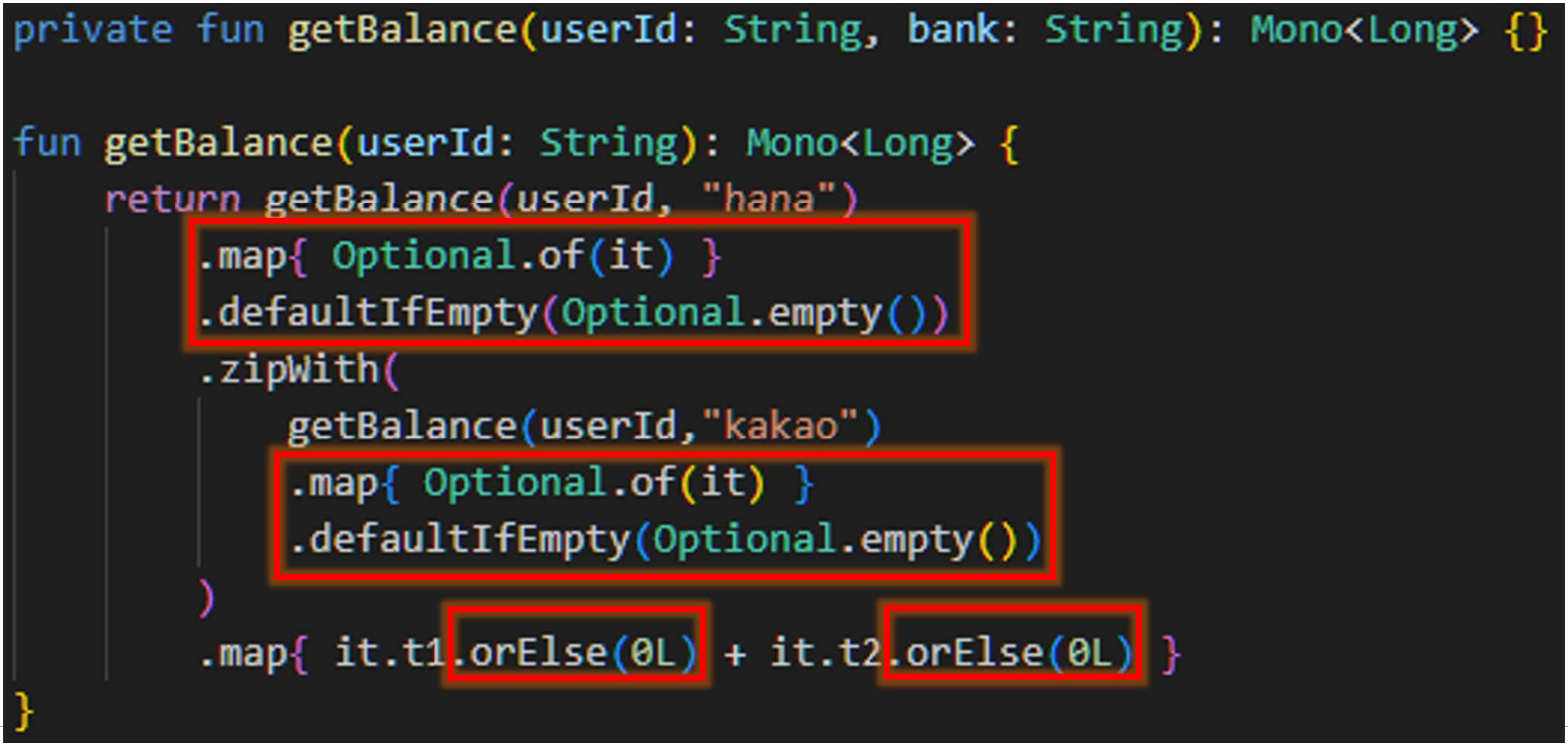
Reactor 구현의 난점



```
private fun getBalance(userId: String, bank: String): Long? {}

fun getBalance(userId: String): Long {
    val a = getBalance(userId, "hana") ?: 0L
    val b = getBalance(userId, "kakao") ?: 0L
    return a + b
}
```

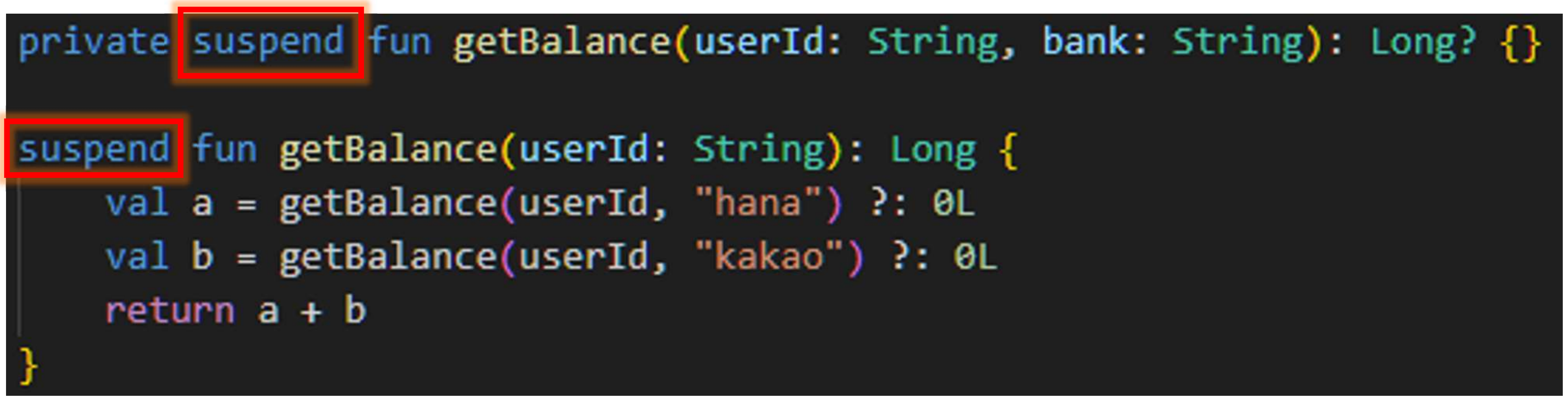
```
private fun getBalance(userId: String, bank: String): Mono<Long> {}

fun getBalance(userId: String): Mono<Long> {
    return getBalance(userId, "hana")
        .zipWith(getBalance(userId,"kakao"))
        .map{ it.t1 + it.t2 }
}
```

Reactor 구현의 난점



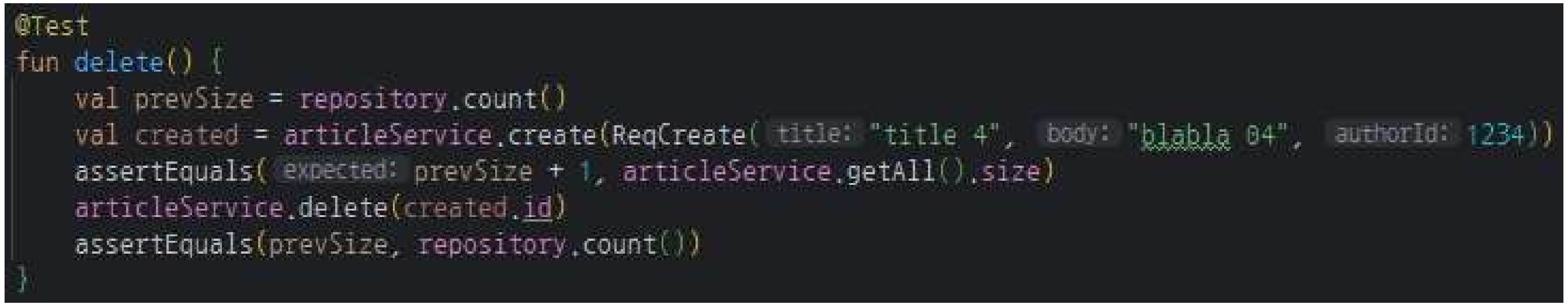
```
private fun getBalance(userId: String, bank: String): Mono<Long> {}

fun getBalance(userId: String): Mono<Long> {
    return getBalance(userId, "hana")
        .map{ Optional.of(it) }
        .defaultIfEmpty(Optional.empty())
        .zipWith(
            getBalance(userId,"kakao")
            .map{ Optional.of(it) }
            .defaultIfEmpty(Optional.empty())
        )
        .map{ it.t1.orElse(0L) + it.t2.orElse(0L) }
}
```

Reactor 구현의 난점

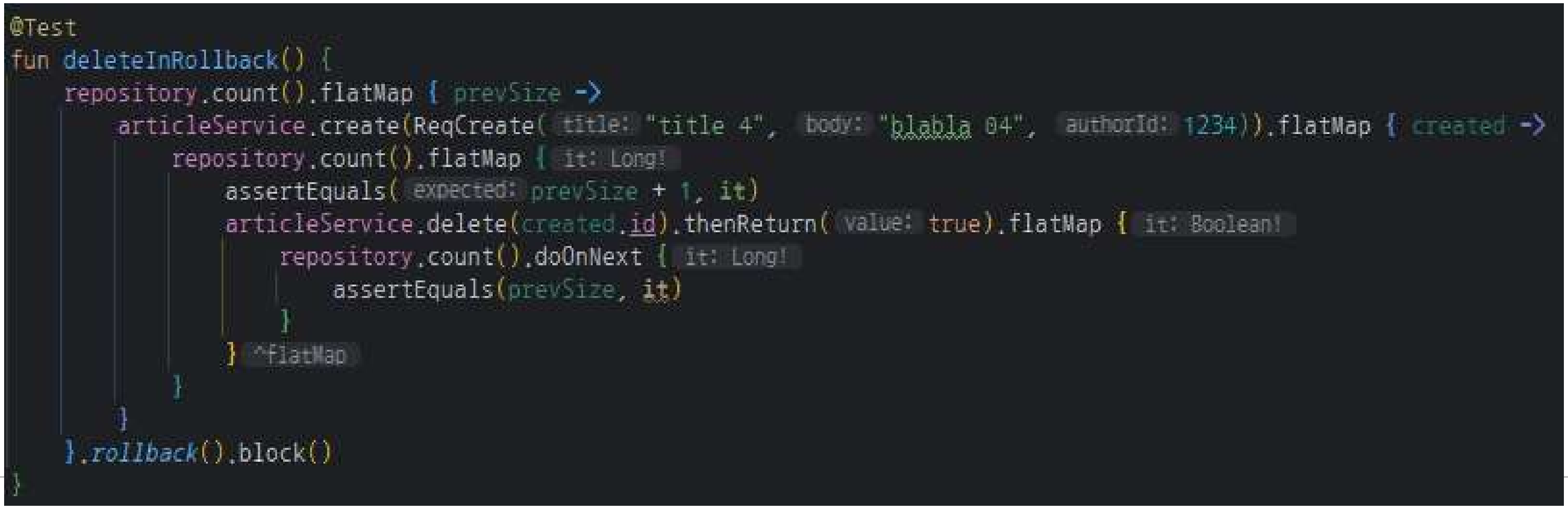


```kotlin
private suspend fun getBalance(userId: String, bank: String): Long? {}

suspend fun getBalance(userId: String): Long {
    val a = getBalance(userId, "hana") ?: 0L
    val b = getBalance(userId, "kakao") ?: 0L
    return a + b
}
```

Reactor coding

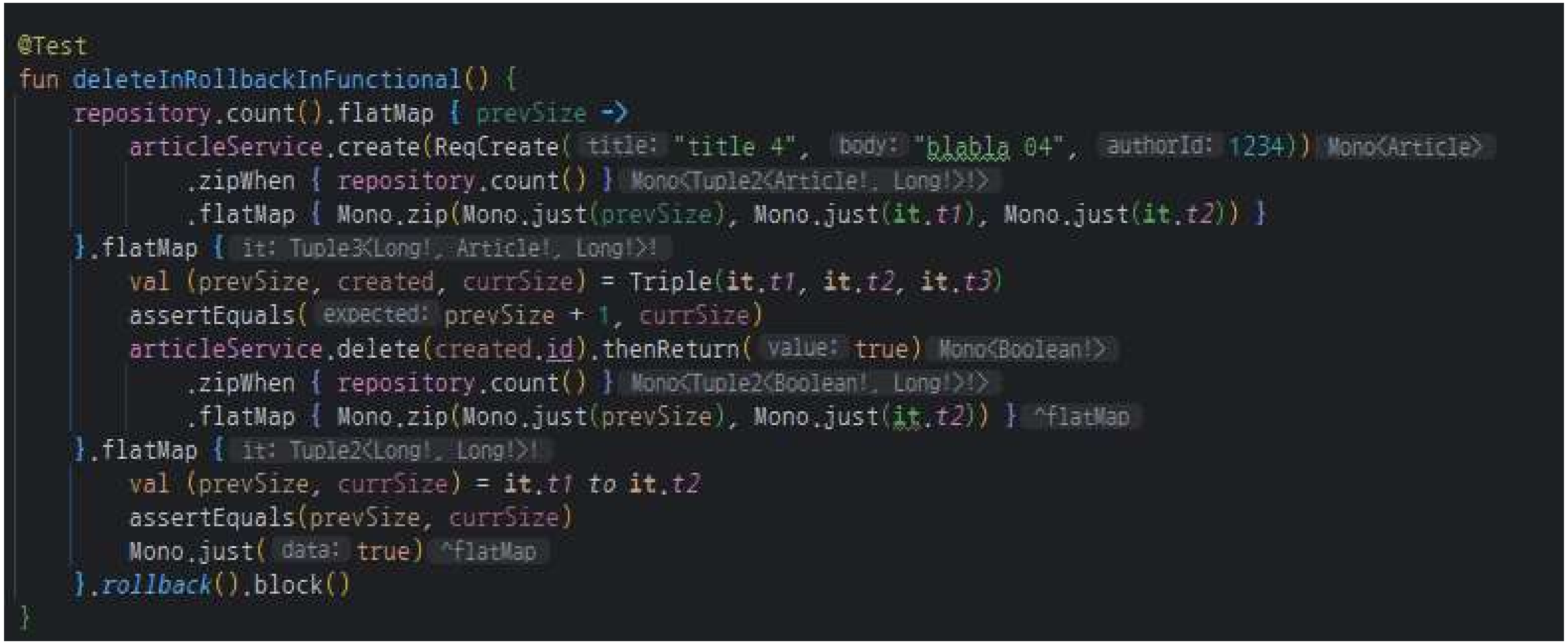


```kotlin
@Test
fun delete() {
    val prevSize = repository.count()
    val created = articleService.create(ReqCreate( title: "title 4",  body: "blabla 04",  authorId: 1234))
    assertEquals( expected: prevSize + 1, articleService.getAll().size)
    articleService.delete(created.id)
    assertEquals(prevSize, repository.count())
}
```



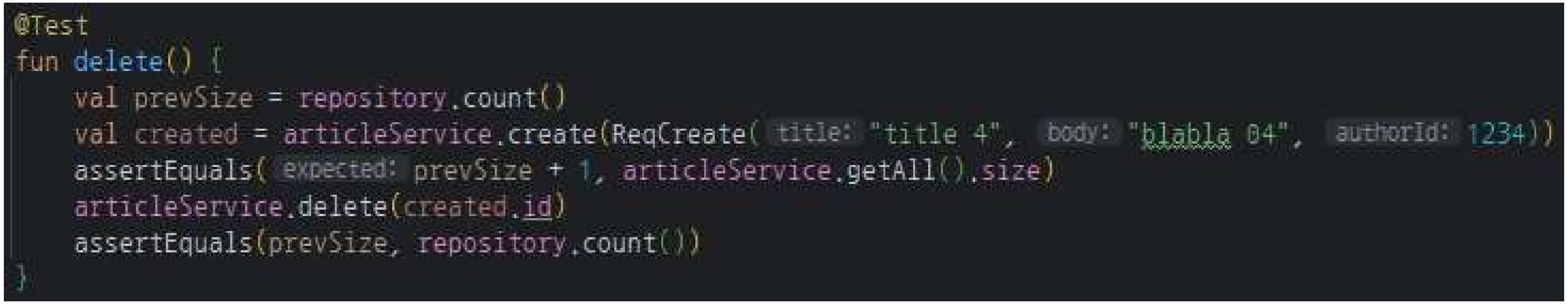
```kotlin
@Test
fun deleteInRollback() {
    repository.count().flatMap { prevSize ->
        articleService.create(ReqCreate( title: "title 4",  body: "blabla 04",  authorId: 1234)).flatMap { created ->
            repository.count().flatMap { it: Long!
                assertEquals( expected: prevSize + 1, it)
                articleService.delete(created.id).thenReturn( value: true).flatMap { it: Boolean!
                    repository.count().doOnNext { it: Long!
                        assertEquals(prevSize, it)
                    }
                } ^flatMap
            }
        }
    }.rollback().block()
}
```

Reactor coding

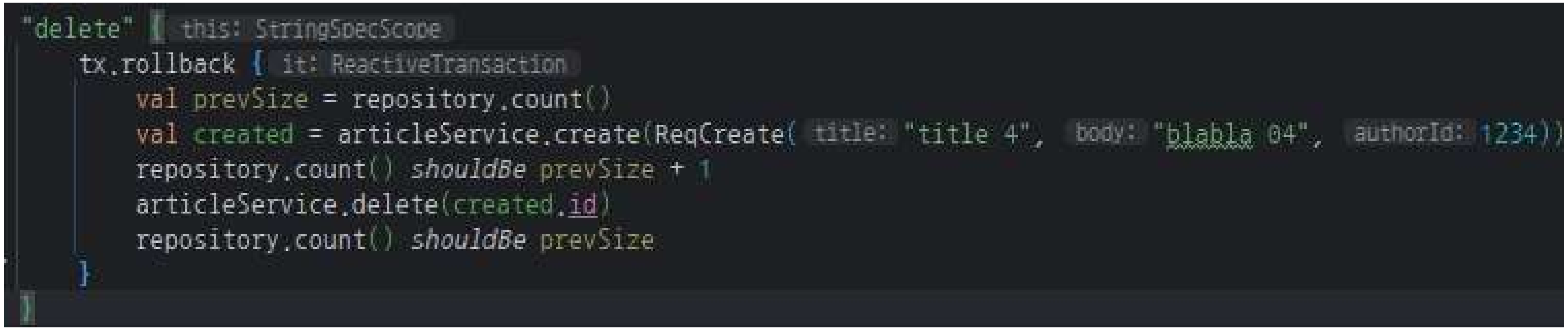


```
@Test
fun deleteInRollbackInFunctional() {
    repository.count().flatMap { prevSize ->
        articleService.create(ReqCreate( title: "title 4",  body: "blabla 04",  authorId: 1234)) Mono<Article>
            .zipWhen { repository.count() } Mono<Tuple2<Article!, Long!>!>
            .flatMap { Mono.zip(Mono.just(prevSize), Mono.just(it.t1), Mono.just(it.t2)) }
    }.flatMap {  it: Tuple3<Long!, Article!, Long!>!
        val (prevSize, created, currSize) = Triple(it.t1, it.t2, it.t3)
        assertEquals( expected: prevSize + 1, currSize)
        articleService.delete(created.id).thenReturn( value: true) Mono<Boolean!>
            .zipWhen { repository.count() } Mono<Tuple2<Boolean!, Long!>!>
            .flatMap { Mono.zip(Mono.just(prevSize), Mono.just(it.t2)) } ^flatMap
    }.flatMap {  it: Tuple2<Long!, Long!>!
        val (prevSize, currSize) = it.t1 to it.t2
        assertEquals(prevSize, currSize)
        Mono.just( data: true) ^flatMap
    }.rollback().block()
}
```

Coroutine coding

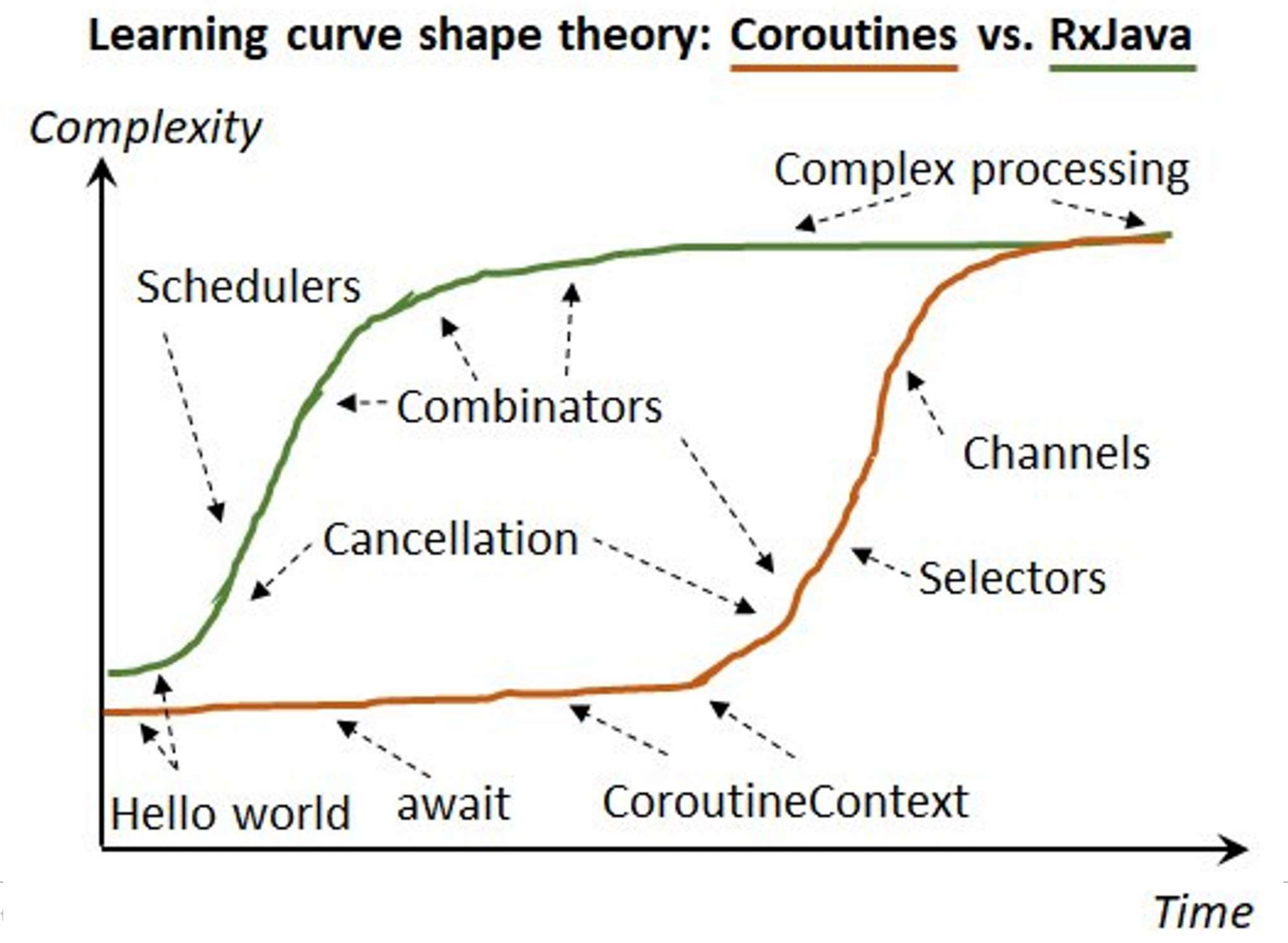


```
@Test
fun delete() {
    val prevSize = repository.count()
    val created = articleService.create(ReqCreate( title: "title 4", body: "blabla 04", authorId: 1234))
    assertEquals( expected: prevSize + 1, articleService.getAll().size)
    articleService.delete(created.id)
    assertEquals(prevSize, repository.count())
}
```



```
"delete" { this: StringSpecScope
    tx.rollback { it: ReactiveTransaction
        val prevSize = repository.count()
        val created = articleService.create(ReqCreate( title: "title 4", body: "blabla 04", authorId: 1234))
        repository.count() shouldBe prevSize + 1
        articleService.delete(created.id)
        repository.count() shouldBe prevSize
    }
}
```

Learning Curve between Reactor and Coroutine



2018.03.30, Davie Karnok

https://twitter.com/akarnokd/status/979732723152687106

Async NIO

CPS Pattern

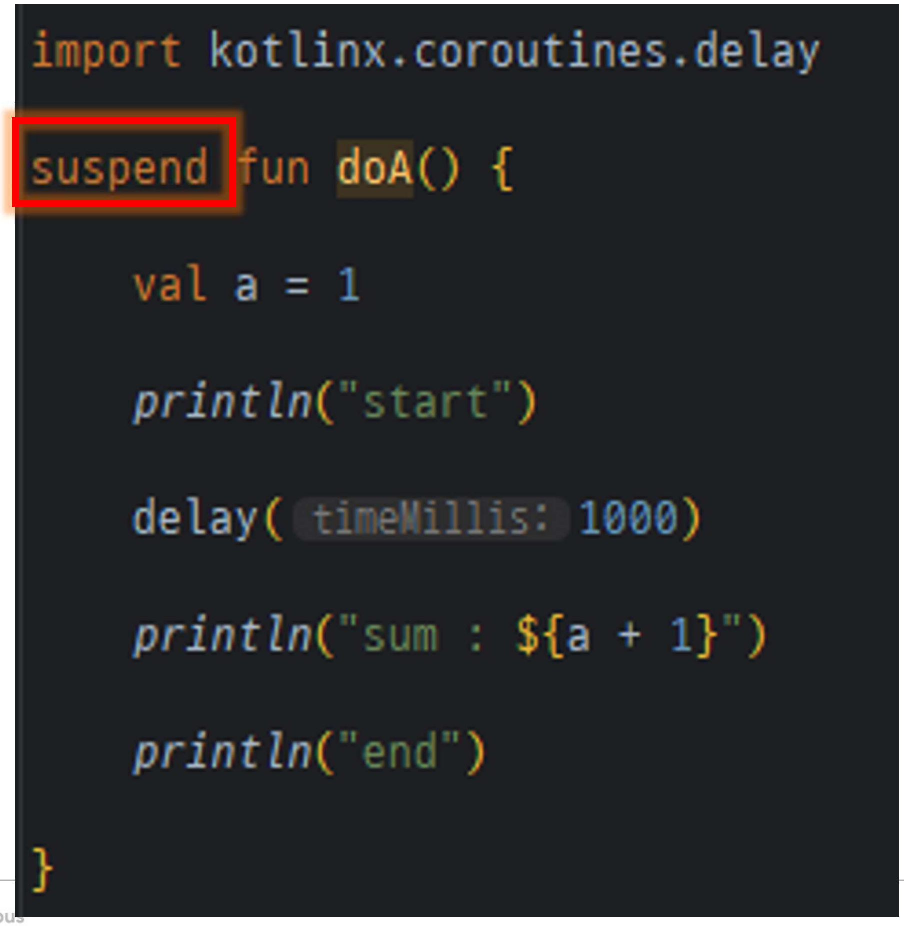
Coroutine 이란 ?

Coroutine

- C++ (C++20~)
  - stackless coroutine
- Rust (2018~)
- GO
  - goroutine
- Javascript
  - async / await

- Kotlin (1.3~)
- PHP (5.5~)
- C# (2.0~)
- Python (3.5~)
- Lua
  - thread

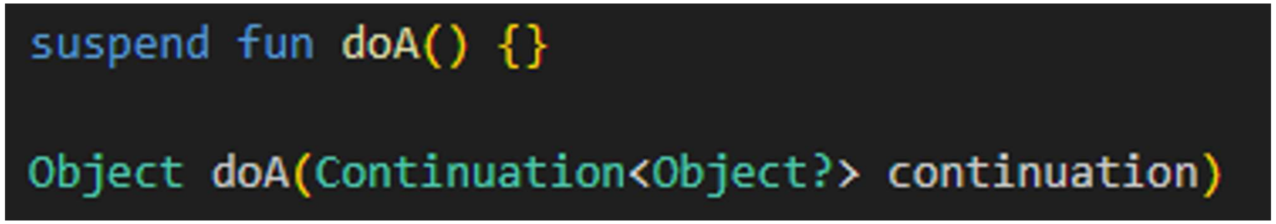
Coroutine 이란 ?



```kotlin
import kotlinx.coroutines.delay

suspend fun doA() {

    val a = 1

    println("start")

    delay( timeMillis: 1000)

    println("sum : ${a + 1}")

    println("end")

}
```

Coroutine 이란 ?

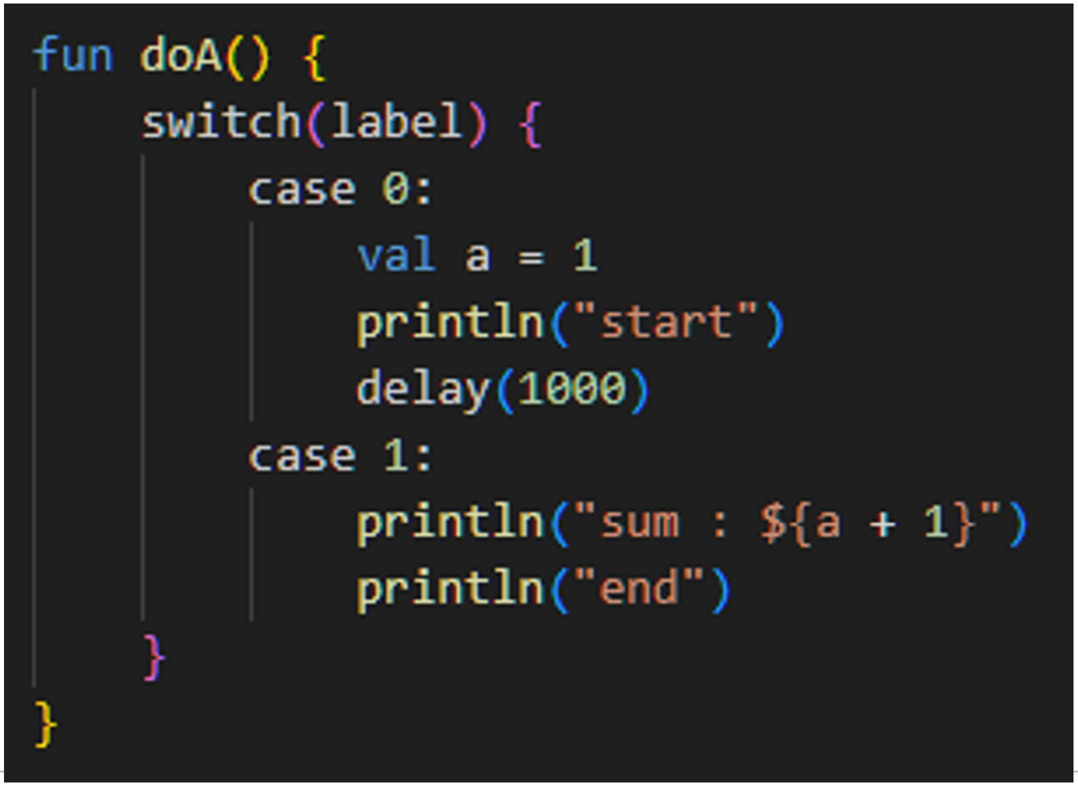
compiled by Kotlin



```
suspend fun doA() {}

Object doA(Continuation<Object?> continuation)
```

Coroutine 이란 ?

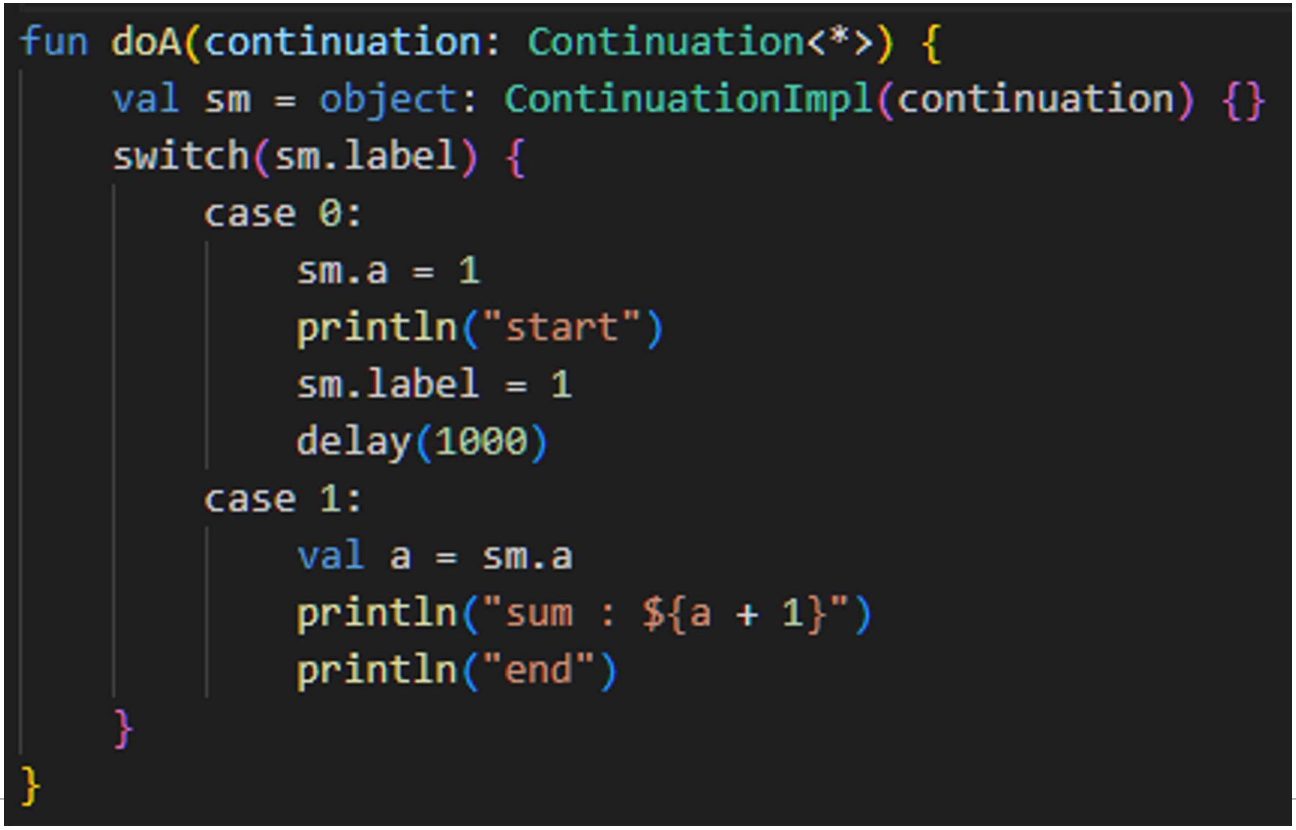
Tagging label



```
fun doA() {
    switch(label) {
        case 0:
            val a = 1
            println("start")
            delay(1000)
        case 1:
            println("sum : ${a + 1}")
            println("end")
    }
}
```

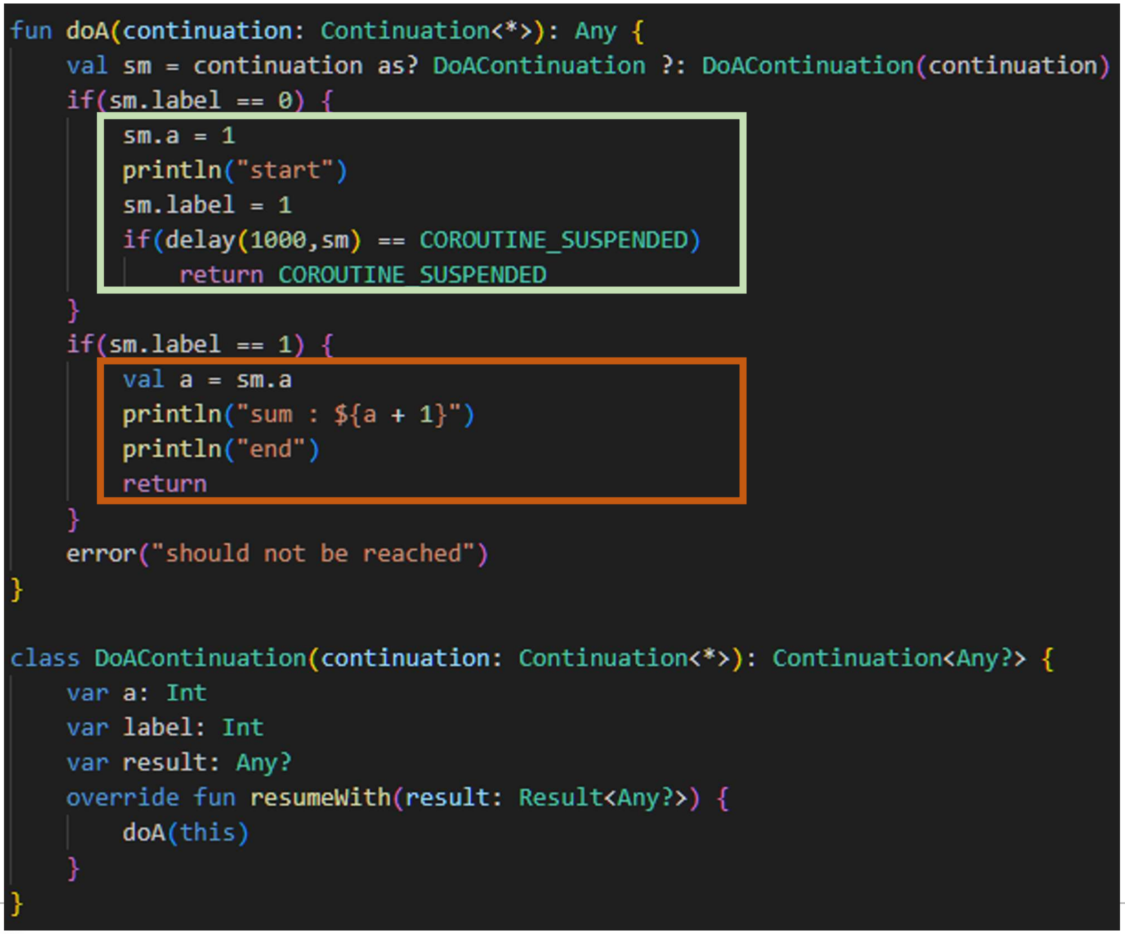
Coroutine 이란 ?

Adding continuation

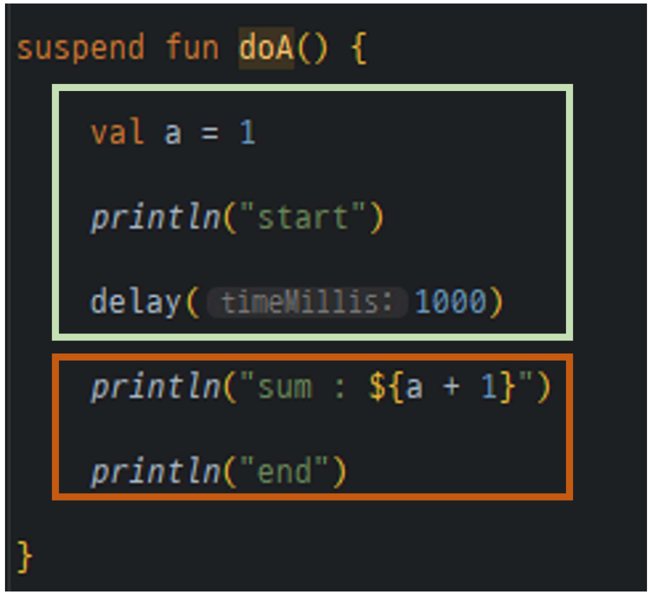


```
fun doA(continuation: Continuation<*>) {
    val sm = object: ContinuationImpl(continuation) {}
    switch(sm.label) {
        case 0:
            sm.a = 1
            println("start")
            sm.label = 1
            delay(1000)
        case 1:
            val a = sm.a
            println("sum : ${a + 1}")
            println("end")
    }
}
```

Coroutine 이란 ?

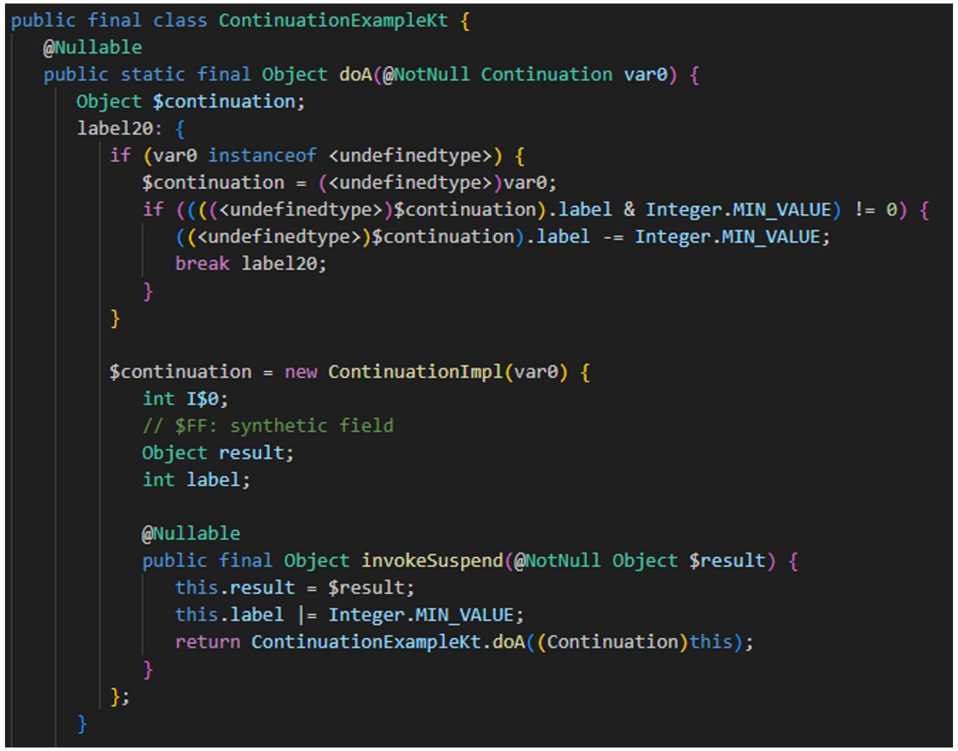


```
fun doA(continuation: Continuation<*>): Any {
    val sm = continuation as? DoAContinuation ?: DoAContinuation(continuation)
    if(sm.label == 0) {
        sm.a = 1
        println("start")
        sm.label = 1
        if(delay(1000,sm) == COROUTINE_SUSPENDED)
            return COROUTINE_SUSPENDED
    }
    if(sm.label == 1) {
        val a = sm.a
        println("sum : ${a + 1}")
        println("end")
        return
    }
    error("should not be reached")
}

class DoAContinuation(continuation: Continuation<*>): Continuation<Any?> {
    var a: Int
    var label: Int
    var result: Any?
    override fun resumeWith(result: Result<Any?>) {
        doA(this)
    }
}
```

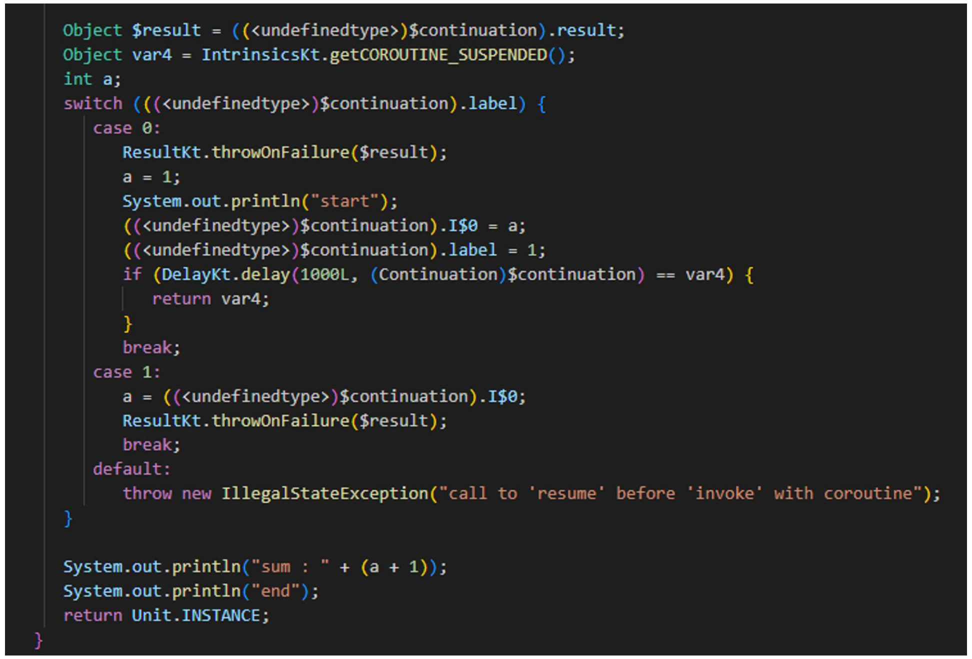


```
suspend fun doA() {
    val a = 1
    println("start")
    delay( timeMillis: 1000)
    println("sum : ${a + 1}")
    println("end")
}
```

Coroutine 이란 ?



```
public final class ContinuationExampleKt {
   @Nullable
   public static final Object doA(@NotNull Continuation var0) {
      Object $continuation;
      label20: {
         if (var0 instanceof <undefinedtype>) {
            $continuation = (<undefinedtype>)var0;
            if ((((<undefinedtype>)$continuation).label & Integer.MIN_VALUE) != 0) {
               ((<undefinedtype>)$continuation).label -= Integer.MIN_VALUE;
               break label20;
            }
         }

         $continuation = new ContinuationImpl(var0) {
            int I$0;
            // $FF: synthetic field
            Object result;
            int label;

            @Nullable
            public final Object invokeSuspend(@NotNull Object $result) {
               this.result = $result;
               this.label |= Integer.MIN_VALUE;
               return ContinuationExampleKt.doA((Continuation)this);
            }
         };
      }
```



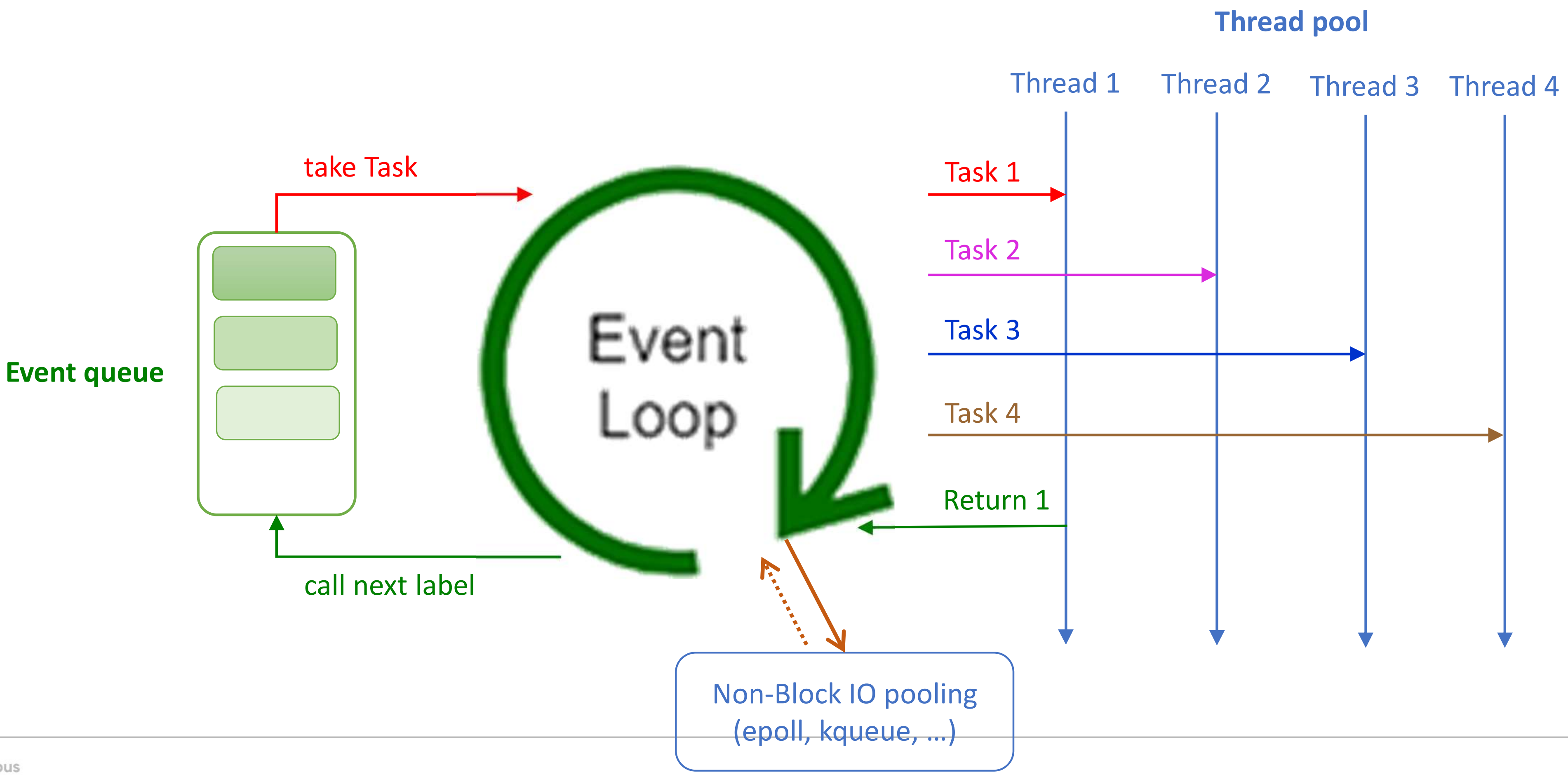
```
         Object $result = ((<undefinedtype>)$continuation).result;
         Object var4 = IntrinsicsKt.getCOROUTINE_SUSPENDED();
         int a;
         switch (((<undefinedtype>)$continuation).label) {
            case 0:
               ResultKt.throwOnFailure($result);
               a = 1;
               System.out.println("start");
               ((<undefinedtype>)$continuation).I$0 = a;
               ((<undefinedtype>)$continuation).label = 1;
               if (DelayKt.delay(1000L, (Continuation)$continuation) == var4) {
                  return var4;
               }
               break;
            case 1:
               a = ((<undefinedtype>)$continuation).I$0;
               ResultKt.throwOnFailure($result);
               break;
            default:
               throw new IllegalStateException("call to 'resume' before 'invoke' with coroutine");
         }

         System.out.println("sum : " + (a + 1));
         System.out.println("end");
         return Unit.INSTANCE;
      }
```

Coroutine Summary

Kotlin Coroutine 의 suspend 함수는

- Kotlin compiler에 의해 CPS 패턴으로 변환
- coroutine dispatcher 에 의해 실행 또는 재개
- suspend 함수는 중단 지점까지 비선점형으로 동작
  - thread는 실행 스케줄이 kernel에 의해 제어됨
- Context는 continuation이라는 parameter 형태로 전달
  - thread context switching 발생하지 않음

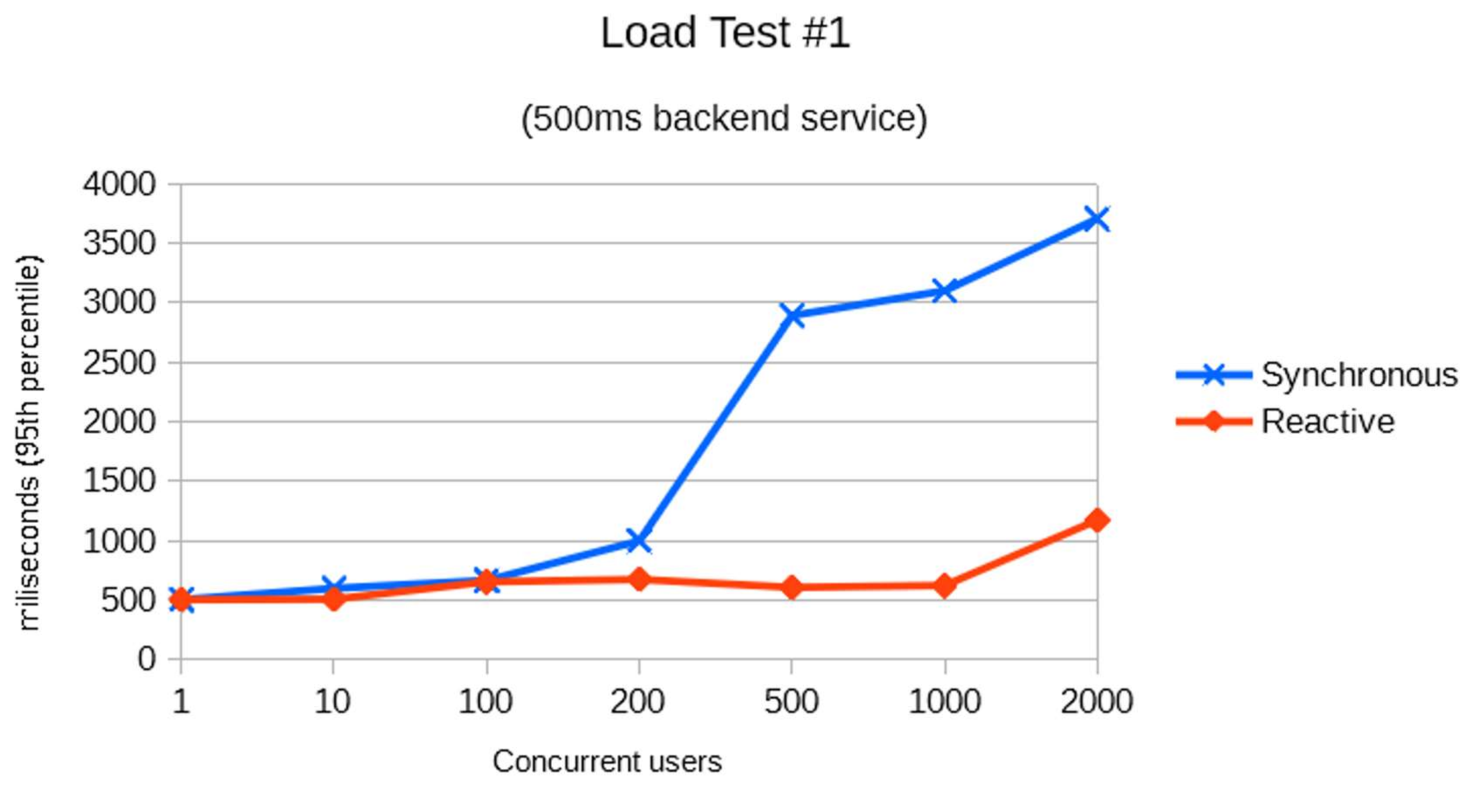
Event Loop


Thread pool
Thread 1
Thread 2
Thread 3
Thread 4
take Task
Event queue
Event Loop
Task 1
Task 2
Task 3
Task 4
Return 1
call next label
Non-Block IO pooling (epoll, kqueue, ...)

Async NIO

Pros / Cons

성능 비교



https://dzone.com/articles/spring-boot-20-webflux-reactive-performance-test

Spring Webflux 단점

MVC보다 느릴 수 있음

- 적은 리소스로 많은 트래픽을 감당하는 개념

구현 난이도가 높음

- 사소한 Blocking 코드가 전체 처리속도에 악영향을 미칠 수 있음