

Kotlin 함수와 람다식

박연종

20210463


park@duck.com

2024.09.11.

SSL SEMINAR





-  **JETBRAINS**에서 2011년 공개한 오픈 소스 프로그래밍 언어
- 개발 당시 JetBrains R&D 센터가 위치했던 러시아의 코틀린 섬에서 이름을 차용
- JVM 기반으로 애플리케이션 개발 시 Java와 혼용해 사용할 수 있음
- 불필요한 세미콜론이나 타입 선언 등을 줄여 코드의 가독성을 높임
- 2017년 구글이 안드로이드 공식 언어로 추가하면서 인기가 많아짐
- 당근, 에이블리, 토스뱅크, 무신사 등에서 백엔드, 안드로이드 앱 개발 시 사용하고 있음



```
1 리턴타입 · 함수명(파라미터타입 · 파라미터명) · {↵  
2   ····return 값;↵  
3 }
```



```
1 fun 함수명(파라미터명 : 파라미터타입) : 리턴타입 · {↵  
2   ····return 값↵  
3 }
```

```
1 리턴타입 · 함수명(파라미터타입 · 파라미터명) · {  
2   ...return 값;  
3 }
```

```
1 fun 함수명(파라미터명 : 파라미터타입) : 리턴타입 {  
2   ...return 값  
3 }
```

/*

형식	자료형	크기
정수형	Byte	1 Byte
	Short	2 Byte
	Int	4 Byte
	Long	8 Byte
부호 없는 정수형	UByte	1 Byte
	UShort	2 Byte
	UInt	4 Byte
	ULong	8 Byte
실수형	Float	4 Byte
	Double	8 Byte
논리형	Boolean	1 Bit
문자형	Char	2 Byte
문자열	String	

*/



```
1 public class SystemSoftwareLab {  
2     ... public static void main(String[] args) {  
3         ... // Fill in the code.  
4     }  
5 }
```



```
1 fun main() {  
2     ... // Fill in the code.  
3 }
```



```
1 fun sum(a: Int, b: Int): Int {  
2     ...return a + b  
3 }
```



```
1 fun sum(a: Int, b: Int): Int = a + b
```



```
1 fun sum(a: Int, b: Int) = a + b
```



```
1 int sumAll(int ... a) {  
2     int res = 0;  
3     for (int j : a) {  
4         res += j;  
5     }  
6     return res;  
7 }
```



```
1 fun sumAll(vararg n: Int): Int {  
2     var res = 0  
3     for (j: Int in n) {  
4         res += j  
5     }  
6     return res  
7 }
```



```
1 fun main() {  
2     println(sumAll(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))  
3 }
```



```
1 fun main() {  
2     val nums = intArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
3     println(sumAll(*nums))  
4 }
```




```
1 fun sum(  
2     a: Int = 0,  
3     b: Int = 0  
4 ) = a + b
```



```
1 fun sum(↵  
2   ... a: Int = 0, ↵  
3   ... b: Int = 0 ↵  
4 ) = a + b
```



```
1 fun main() {↵  
2   ... println(sum(b = 3)) ↵  
3 }
```

1급 객체 (First-class citizen_[,type,object,entity,value])

- 공정한 대우를 받는 집단의 구성원
- Christopher Strachey에 의해 고안되어 Robin Popplestone의 정의가 널리 퍼짐
- 모든 1급 객체는 함수의 실질적인 매개변수가 될 수 있다.
- 모든 1급 객체는 함수의 반환 값이 될 수 있다.
- 모든 1급 객체는 할당의 대상이 될 수 있다.
- 모든 1급 객체는 비교 연산(==, equal)을 적용할 수 있다.
- 함수가 1급 객체인 프로그래밍 언어는 Dart, Kotlin, Swift, Python, JavaScript, Rust 등이 있음
- Java에서는 람다식을 통해 메서드가 1급 객체로 다뤄짐

1급 객체 (First-class citizen_[,type,object,entity,value])

- 모든 1급 객체는 함수의 실질적인 매개변수가 될 수 있다.

```
1 fun operateOnNumbers(a: Int, b: Int, operation: (Int, Int) → Int): Int {  
2     ...return operation(a, b)  
3 }  
4  
5 fun main() {  
6     ...val sum = operateOnNumbers(5, 3, {x, y → x + y})  
7     ...println(sum) // 출력: 8  
8 }
```

1급 객체 (First-class citizen_[,type,object,entity,value])

- 모든 1급 객체는 함수의 반환 값이 될 수 있다.

```
1 fun operateOnNumbers(a: Int, b: Int, operation: (Int, Int) → Int): Int {  
2     ...return operation(a, b)  
3 }  
4  
5 fun main() {  
6     ...val sum = operateOnNumbers(5, 3, { x, y → x + y })  
7     ...println(sum) // 출력: 8  
8 }
```

1급 객체 (First-class citizen_[,type,object,entity,value])

- 모든 1급 객체는 할당의 대상이 될 수 있다.

```
1 fun main() {  
2     val s = ::sum  
3     println(s(10, 5))  
4 }  
5  
6 fun sum(a: Int, b: Int): Int = a + b
```

1급 객체 (First-class citizen_[,type,object,entity,value])

- 모든 1급 객체는 비교 연산(==, equal)을 적용할 수 있다.

```
1 fun main() {
2     val hi3 = printlnRepeat(3, "Hi")
3     val hi3Two = printlnRepeat(3, "Hi")
4     val hi3D = hi3
5
6     println(hi3 == hi3Two)
7     println(hi3 == hi3D)
8 }
9
10 fun printlnRepeat(repeat: Int, str: String): () → Unit {
11     return {
12         for (i in 1 .. repeat) {
13             println(str)
14         }
15     }
16 }
```

```
1 /* [Result]
2  * false
3  * true
4  */
```

```
1 fun main() {  
2     val lambdaExpressionName: (String) → Unit = { name → println(name) }  
3     val lambdaExpressionName2: (String) → Unit = { name: String → println(name) }  
4  
5     fun lambdaExpressionName3(name: String): Unit {  
6         println(name)  
7     }  
8     fun lambdaExpressionName4(name: String) {  
9         println(name)  
10    }  
11  
12    val lambdaExpressionName5: () → Unit = { println("PARK, Yeonjong") }  
13    val lambdaExpressionName6 = { println("PARK, Yeonjong") }  
14 }
```




```
1 fun main() {  
2     val arr = arrayOf("Apple", "Banana", "Grape", "PineApple")  
3  
4     arr.forEach({ item → println(item) })  
5  
6     arr.forEach({ println(it) })  
7 }
```

후행 람다 (Trailing Lambda)



```
1 fun main() {  
2     val arr = arrayOf("Apple", "Banana", "Grape", "PineApple")  
3  
4     arr.forEach { item → println(item) }  
5  
6     arr.forEach { println(it) }  
7 }
```



```
1 public inline fun <T> Array<out T>.forEach(action: (T) → Unit): Unit {  
2     for (element in this) action(element)  
3 }
```

```
1 typealias StrPUniR = (String) -> Unit
2
3 fun main() {
4     ... val lambdaExpression: StrPUniR = { println(it) }
5     ... // val lambdaExpression: (String) -> Unit = { message: String ->
6     ... // ... println(message)
7     ... // }
8     ... lambdaExpression("PARK, Yeonjong")
9 }
```

감사합니다

Email / park@duck.com

Insta / [@yeonjong.park](https://www.instagram.com/yeonjong.park)

GitHub / [patulus](https://github.com/patulus)

