

Chapter. 01

Kotlin

| Kotlin으로 개발하기 vs Java로 개발하기

FAST CAMPUS
ONLINE
Kotlin Android |

강사. 강경완

I Null Safe

```
Integer a = 100;
```

```
val b: Int? = 100
```

```
val c: Int = 100
```

```
a = null;
```

```
/// 중략 ///
```

```
a.sum(); // NullPointerException 이 날 수도 있음.
```

```
// null safe 한 코드를 구성해야함
```

```
if (a != null) {
```

```
    a.sum();
```

```
}
```

```
b?.sum() // null 일 경우 실행하지 않음.
```

```
c.sum() // 애초에 nullable 함
```

I Scope Function (apply, with, let, also, run)

Function selection

To help you choose the right scope function for your purpose, we provide the table of key differences between them.

Function	Object reference	Return value	Is extension function
<code>let</code>	<code>it</code>	Lambda result	Yes
<code>run</code>	<code>this</code>	Lambda result	Yes
<code>run</code>	-	Lambda result	No: called without the context object
<code>with</code>	<code>this</code>	Lambda result	No: takes the context object as an argument.
<code>apply</code>	<code>this</code>	Context object	Yes
<code>also</code>	<code>it</code>	Context object	Yes

<https://kotlinlang.org/docs/scope-functions.html#function-selection>

I Scope Function (apply, with, let, also, run)

Apply 함수

```
val person = Person().apply {  
    firstName = "Fast"  
    lastName = "Campus"  
}
```

```
Person person = new Person();  
person.firstName = "Fast";  
person.lastName = "Campus";
```

<https://kotlinlang.org/docs/lambdas.html#function-literals-with-receiver>

I Scope Function (apply, with, let, also, run)

Also 함수

```
Random.nextInt(100).also {  
    print("getRandomInt() generated value $it")  
}
```

```
Random.nextInt(100).also { value ->  
    print("getRandomInt() generated value $value")  
}
```

```
int value = Random().nextInt(100);  
System.out.print(value);
```

I Scope Function (apply, with, let, also, run)

Let 함수

```
val number: Int?  
  
val sumNumberStr = number?.let {  
    "${sum(10, it)}"  
}
```

```
Integer number = null;  
String sumNumberStr = null ;  
  
if (number != null) {  
    sumNumberStr = "" + sum(10, number);  
}
```

I Scope Function (apply, with, let, also, run)

Let 함수

```
val number: Int?  
  
val sumNumberStr = number?.let {  
    "${sum(10, it)}"  
}.orEmpty()
```

```
Integer number = null;  
String sumNumberStr = null;  
  
if (number != null) {  
    sumNumberStr = "" + sum(10, number);  
} else {  
    sumNumberStr = "";  
}
```

I Scope Function (apply, with, let, also, run)

With 함수

```
val person = Person()  
  
with(person) {  
    work()  
    sleep()  
    println(age)  
}
```

```
Person person = new Person();  
  
person.work();  
person.sleep();  
System.out.println(person.age);
```


I Scope Function (apply, with, let, also, run)

Run 함수

```
val result = service.run {  
    port = 8080  
    query()  
}
```

```
service.port = 8080  
Result result = service.query()
```

I Data Class

```
public class JavaObject {  
  
    private String s;  
  
    JavaObject(String s) {  
        this.s = s;  
    }  
  
    public String getS() {  
        return s;  
    }  
  
    public void setS(String s) {  
        this.s = s;  
    }  
  
    // copy  
    // toString  
    // hashCode 등등 생략  
  
}
```

```
data class JavaObject(val s: String)
```

I Lambda expression

```
button.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        '''  
    }  
})
```

```
button.setOnClickListener { v ->  
  
}
```

I lateinit, lazy init

NullSafe 한 코드를 사용하기 위해서 non-null Type 으로 변수를 선언함

초기값이 없는 변수는 어떻게 초기화를 해야할까?
초기값이 없으면 변수 선언 자체가 안되는데...

```
var nullableNumber: Int? = null

lateinit var lateinitNumber: Int

// 추후 초기화하는 코드
lateinitNumber = 10


// 사용할 때
nullableNumber?.add()


lateinitNumber.add()
```

I lateinit, lazy init


NullSafe 한 코드를 사용하기 위해서 non-null Type 으로 변수를 선언함
변수는 미리 선언해놓고 사용할 때 할당해주면 안될까?


```
val lazyNumber :Int by lazy {  
    100  
}  
  
// 사용하기 전까지는 lazyNumber 라는 변수에 100 이 할당되지 않음.  
  
lazyNumber.add()  
// 사용할 때 100이 할당됨
```

 Kotlin v1.4.30

SolutionsDocsCommunityTeachPlay ↗


A modern programming language that makes developers happier.

[Get started](#)[Why Kotlin](#) Developed by [JetBrains](#) & Open-source [Contributors](#)




Multiplatform Mobile

The natural way to share code between mobile platforms




Server-side

Modern development experience with familiar JVM technology



Web Frontend

Extend your projects to web



Android

Recommended by Google for building Android apps

<https://kotlinlang.org/>