Kotlin extension functions

Extension functions and receiver type

Extension functions

Q: Why extension functions if topic was receivers?

A: That is because receiver type is just one term inside the extension function?

Q: Okay, then what is extension function?

A: It is a syntax sugar for the functions that accepts an object. (Show Example 1)

Extension functions

Q: And then how we can call extension function?

A: We apply extension functions onto instances of the class that we extended (Show Example 2)

Q: Okay.. But stop, in Kotlin classes are closed to be extended (inherited) by default, does it mean that extension functions break this restriction?

A: No, they are not members (methods) of the class, they are just beautiful form of the function that accept objects as arguments. (Show Example 3)

Receiver type

Q: Okay, finally, WHAT IS THE RECEIVER TYPE?

A: The **receiver type** is just the type on which extension function was defined. Thus if you have

"fun String.isValidEmail(): Boolean" then String is the receiver type

"*fun User.calculateAge(): Boolean*" then User is the receiver type

Function types

Functions in Kotlin are different from methods in Java. While in Java any "functions" are actually methods of the classes or implementations of functional interfaces, in Kotlin functions are separate part of the language. They are not directly bound to classes, all functions have its own types. These types actually represented by the shapes of functions signatures. Don't worry, let's see Example 4.

Lambas

I think all of you know about lambdas. Almost all languages have some mechanism to have anonymous functions, so I am sure you understand the general idea.

And of course in the strict typed language like Kotlin, to accepts or return such kind of function we need to define the type. But hopefully we already know the function types in Kotlin from the previous slide.

(Show example 5)

Lambda extension function (receiver lambda)

- Let's have a look at the a bit more complex usage of lambda
- And then how we can simplify it with **receiver lambda**

Extension property

And just for you to know, Kotlin also allows extension properties.

See Example7

Extra info

- One of the most important usage of extension functions is to add **inline function** to the interface.
- Shortly **inline functions** are used for two purposes: effectively execute lambdas and reifying generic types. First one is out of the topic, and second one, sometimes can really decrease the code. Let's have a look at Example 8 and Example 9

Extra info

Generic extension function

(See example 10)

Advanced usage of receivers (DSL)

(See example 11)