Leveraging the Power of Kotlin



Sriyank Siddhartha
AUTHOR
linkedin.com/in/sriyank
sriyank.siddhartha@gmail.com



Kotlin Power

Extension Function

Default Parameters

Lambda Expressions

High Order Functions



```
fun display(x) {
    print(x)
}
1. { x -> print(x) }
2. { (x, y) -> print(x + y) }
3. { addNumbers() }
```

Lambda Expression: A function without name

A lambda expression is always surrounded by curly brackets

Its parameters (if any) are declared before ->

The body(if any) goes after ->



```
// Somewhere in class..
playTurn( { rollDice() }, "Player one turn" )
playTurn( { rollDiceTwice() }, "Player one bonus turn" )

fun playTurn(myFunc: () -> Unit, msg: String) { // Higher-Order Function
    myFunc() // rollDice() // rollDiceTwice()
}
```

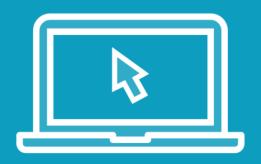
Higher-Order Function

A function that accepts function as parameter

Or A function that returns a function



Demo



Extensions Functions

Default Parameters

Lambda Expressions

Higher-Order Functions



Inline Functions

Higher-Order Functions

- Disadvantages
 - The passed functions are stored as objects
 - Too much of its usage can affect memory and thus performance

- Makes affective use of memory
- Passed functions are no longer stored as objects
- The passed function expands at the call site thus reducing the call overhead



- Too much of its usage increases byte code
- Use it only for short methods or logical code
- Do not use it for large methods

