

Inline Functions



Inline Functions

Make the compiler insert the body of a function at every call site

```
inline fun List<Product>.applyDiscountFast(percentage: Double, operation: (Product) -> Unit) {  
    for (product in this) {  
        product.price *= (1 - percentage / 100)  
        operation(product)  
    }  
}
```

```
products.applyDiscountFast(10.0) { product ->  
    println("${product.name} is now ${product.price} USD")  
}  
  
// call is rewritten (= inlined) to:  
for (product in products) {  
    product.price *= (1 - 10.0 / 100)  
    println("${product.name} is now ${product.price} USD")  
}
```

Inline Functions

Benefits

- compiler-generated code with no overhead (function call, lambda instantiation)
- potential performance benefits for small functions that are repeated a lot in your codebase

Caveats

- potential side effects
- all lambdas are inlined as well, cannot be passed as function values

Sometimes useful to not inline some lambdas – use `noinline`

```
inline fun performOperation(  
    noinline storeOperation: () -> Unit,  
    executeOperation: () -> Unit  
) {  
    // store noinline lambdas as regular values  
    GlobalStore.store(storeOperation)  
  
    // execute the other  
    executeOperation()  
}
```

Kotlin rocks

