AVISI

FP and OOP: Best Friends Forever?

Combining the best of two worlds

Agenda.

- → What?
- → So what?
- → Now what?



What?

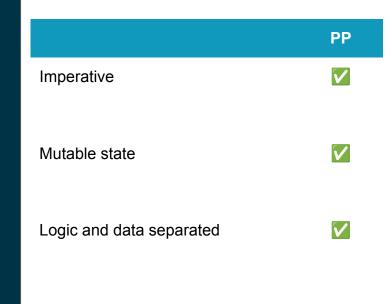
Setting the stage



Paradigms.

- → Procedural (PP)
- → Object-Oriented (OOP)
- → Functional (FP)

Procedural Programming.





Object-Oriented Programming.

	PP	ООР
Imperative	V	V
Mutable state	V	V
Logic and data separated	V	
Encapsulation / data hiding		V
Dynamic polymorphism		V



Functional Programming.

	PP	ООР	FP
Imperative	V	V	
Declarative			V
Mutable state	V	V	
Referential transparency			V
Logic and data separated	V		V
Encapsulation / data hiding		V	
Dynamic polymorphism		V	
Functions are first-class citizens			V



Referential Transparency?

The result of a function depends **only** on its input parameters.

```
fun transparent(a: Int, b: Int) =
    a + b

fun opaque(a: Int, b: Int) =
    a + b + Random.nextInt()
```

First-class functions?

Functions are **values**, just like integers, strings, etc.

```
val list = listOf("hello", "world")

val toUppercase: (String) -> String =
    { s: String -> s.uppercase() }

println(
    list.map(toUppercase) // declarative!
)

// Output: [HELLO, WORLD]
```

So what?



We can use FP in OOP languages!

Many modern OOP languages have adopted FP concepts

FunctionalObject Oriented Programming.

	PP	ООР	FP	F-OOP
Imperative	V	V		
Declarative			V	
Mutable state	V	V		**
Referential transparency			V	6
Logic and data separated	V		V	
Encapsulation / data hiding		V		
Dynamic polymorphism		V		
Functions as first class citizens			V	



Demo!



Now what?



OOP and FP: BFFs?

Yes!

But a good relationship is about giving and receiving

FP

- → Allow some impure code
- → + Encapsulation

OOP

- → Constructors
- → Referential transparency
- → + More patterns
- → + First-class functions