JAVA GOES MODERN

GOING BEYOND STREAMING-API

- Christoph Welcz
- @ChristophWelcz
- github.com/enolive/java-goes-modern

Software Craftsmanship Community @ DATEV

Welcome to FP Club!

Rule #1: You do not talk about the m-word

Rule #2: You DO NOT talk about the m-word

WHY'S FP SO POPULAR?

Reactive programming
Microservices Cloud
Expressiveness Parallell
computing Multithreading FP languages
filter/map/reduce Promises/Observables
Stateless Immutability Lazy
Evaluation Avoid Nesting

JAVA 8 ALLOWS FP!

- Streaming API
- Functional Interfaces
- Optional Type

Java

```java
List<String> list = Arrays.asList(
    null, "anna", "bernd", "chris", "");

String result = list
    .stream()
    .filter(word -> word != null)
    .filter(word -> !word.isEmpty())
    .map(String::toUpperCase)
    .collect(Collectors.joining(", "));
// result = "ANNA, BERND, CHRIS";
```

Java

```java
String input = ...;

String result = Optional
     .ofNullable(input)
     .filter(i -> !i.isEmpty())
     .orElseGet(() -> "default value");
// result = "default value"; when input is null or empty
// result = input; otherwise
```

COLLECT TOO COMPLICATED

- allows converting to Map, List, Group, Set, String, ...
- violates SRP 😟

Java

```java
String sentence = ...;

Map<Character, Long> countChars = sentence
    .chars()
    .boxed()
    .map(i -> (char)i.intValue())
    .collect(Collectors.groupingBy(
        Function.identity(), Collectors.counting()));
```

STANDARD-API TOO LIMITED

Either Try Lazy Tuple Applicative

Validation Functions > 2 parameters

ordered reduce/fold unfold indexed

operations reverse cycle

append/prepend partial application

currying zipping memoization

- pronounce as "**waver**"
- first version 2014
- heavily inspired by Scala
- FP collections & data types

LET'S START CODING...

INTERESTING STUFF

- 🔗 Vavr documenation
- YouTube Venkat Subramanian: Let's get lazy
- YouTube Bodil Stokke: What every Hipster should know about Functional Programming
- 🔗 Marco Emrich: Loops must die! 🇩🇪