Fp-Gilde Session 0

21.02.2022

Was ist FP (der harte Kern)

- Immutability
- Pure Funktionen
- Funktionen as Werte

Immutability

- Alles ist eine Konstante
- Konstante + Änderung = neue Konstante

Pure Funktionen

- Bei gleichen Parametern gleiches Ergebnis
- Reihenfolge des Aufrufs nicht relevant

Reihenfolge

```
// Die Reihenfolge der Afrufe ist (definitiv) nicht relevant
// wenn beide Funktionnen pur sind

valid = validateUser(user);
phone = formatPhoneNumber(user);
```

Funktionen als Werte

```
const add = a => b => a + b
[1,2,3,4].map(add(2))
// [3, 4, 5, 6]
```

Ziele

- Primär: Maximierung von puren Funktionen
- Sekundär: Kapselung von Seiteneffekten

Vorteile

- Testbarkeit
- Kombinierbarkeit
- Paralellisierbarkeit
- Deklarativer Stil

Testbarkeit

Pure Funktionen brauchen keine Mocks

Kombinierbarkeit

(🍋 -> 🍎) + (🍎 -> 🍍)

=

🍋 -> 🍍

https://youtu.be/WhEkBCWpDas

Parallelisierbarkeit

https://github.com/ali-baykal/aoc_2021/blob/main/08/part1.hs

Nachteile

- Speichergebrauch
- Performanz

Performanz / Quicksort in Haskell

https://koerbitz.me/posts/Efficient-Quicksort-in-Haskell.html

Performanz / Lösungen

- Paralellisierbarkeit
- Memoisation
- Immutable Datenstrukturen
- Lazyness

Lazyness (I)

```
add a b = a + b -- Wird nicht ausgeführt
evaluate value = 42
main = do
    print(evaluate(add 1 2))
```

https://youtu.be/F73kB4XZQ4I

Lazyness (II)

```
calc a b =
    | a > b = c + a
    | otherwise = d + b
    where
        c = calc b a
        d = a + 2
```

Lazyness (II)

```
const calc = (a, b) => {
    // const c = calc(b, a); Endlosschleife :(
    const d = a + 2;

    if(a > b) {
        const c = calc(b, a);
        return c + a;
    } else {
        return d + b;
    }
}
```