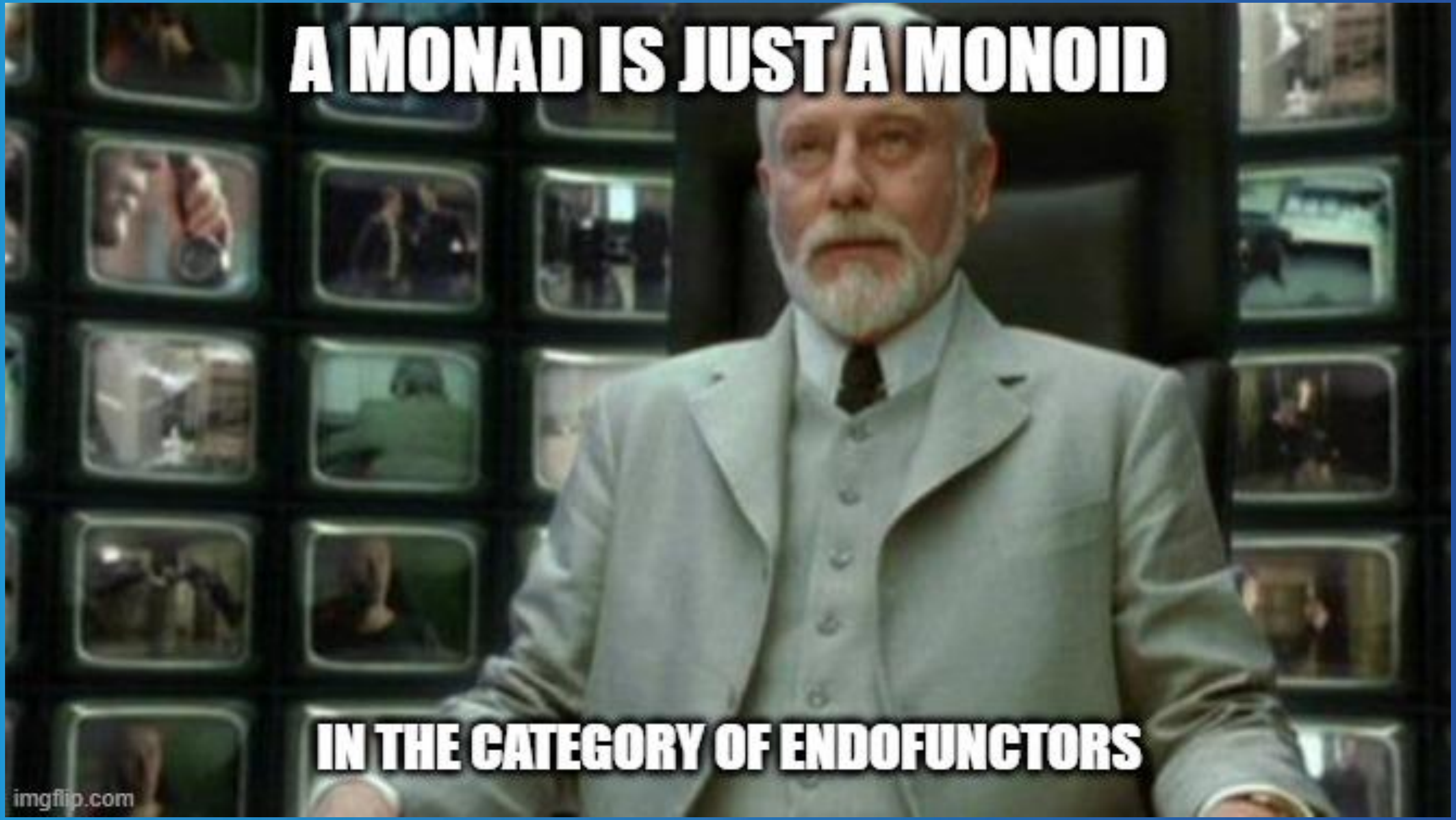
Functional Progamming for mere mortals

Wekoslav Stefanovski
@swekster
https://github.com/sweko
https://www.youtube.com/@swekster

>whoami

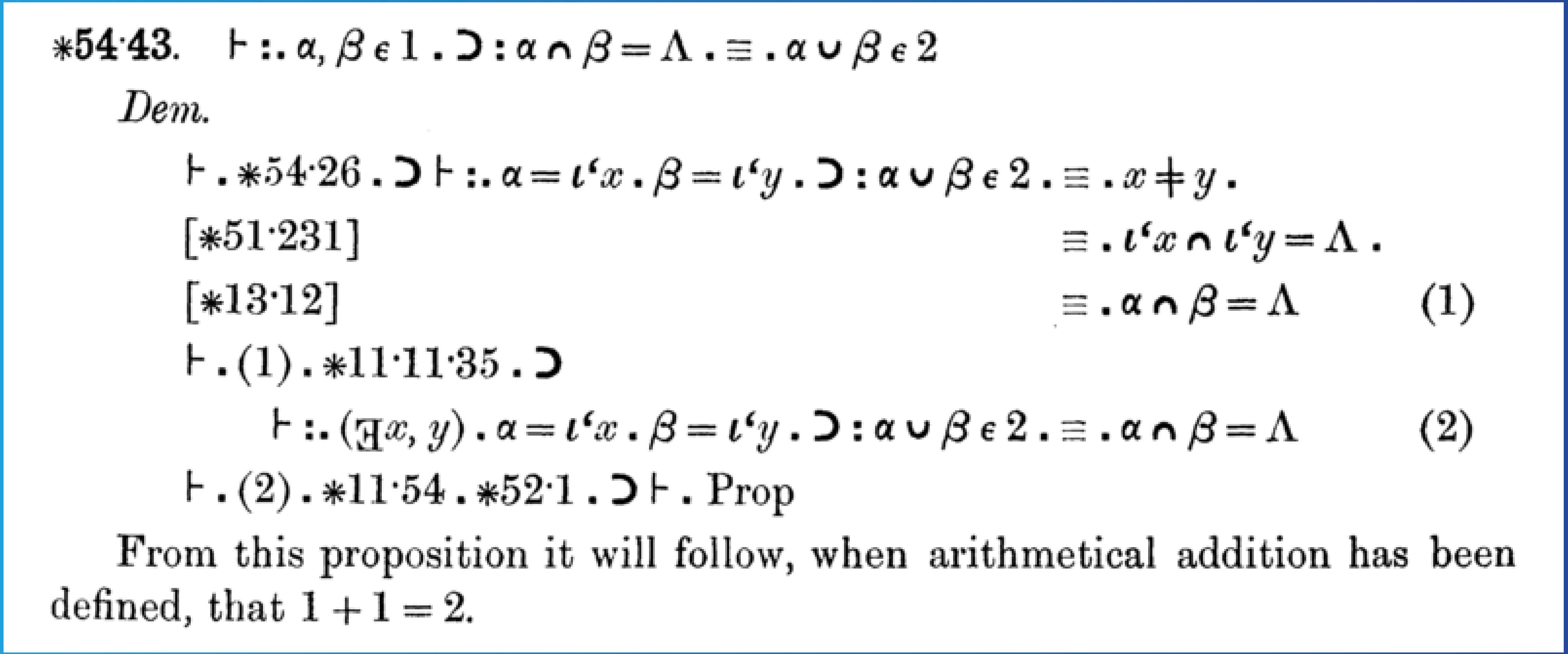
- Head of development at Sourcico, Macedonia
- Professional Memeloper
- Coding professionally since last century
- I love programming, I love programmers
- Long and fruitful love relationship with C#
- Long and fruitful love/hate relationship with JavaScript
- Very lazy, so very few slides (and those are mostly memes)

>functional programming



A programming technique that combines the flexibility and power of abstract mathematics with the intuitive clarity of abstract mathematics.

>functional programming



*54·43. $\vdash :. \alpha, \beta \in 1 . \supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54\cdot26 . \supset \vdash :. \alpha = \iota\text{‘}x . \beta = \iota\text{‘}y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

[*51·231] $\equiv . \iota\text{‘}x \cap \iota\text{‘}y = \Lambda .$

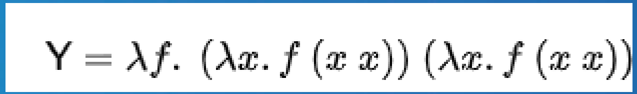
[*13·12] $\equiv . \alpha \cap \beta = \Lambda$ (1)

$\vdash . (1) . *11\cdot11\cdot35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota\text{‘}x . \beta = \iota\text{‘}y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda$ (2)

$\vdash . (2) . *11\cdot54 . *52\cdot1 . \supset \vdash .$ Prop

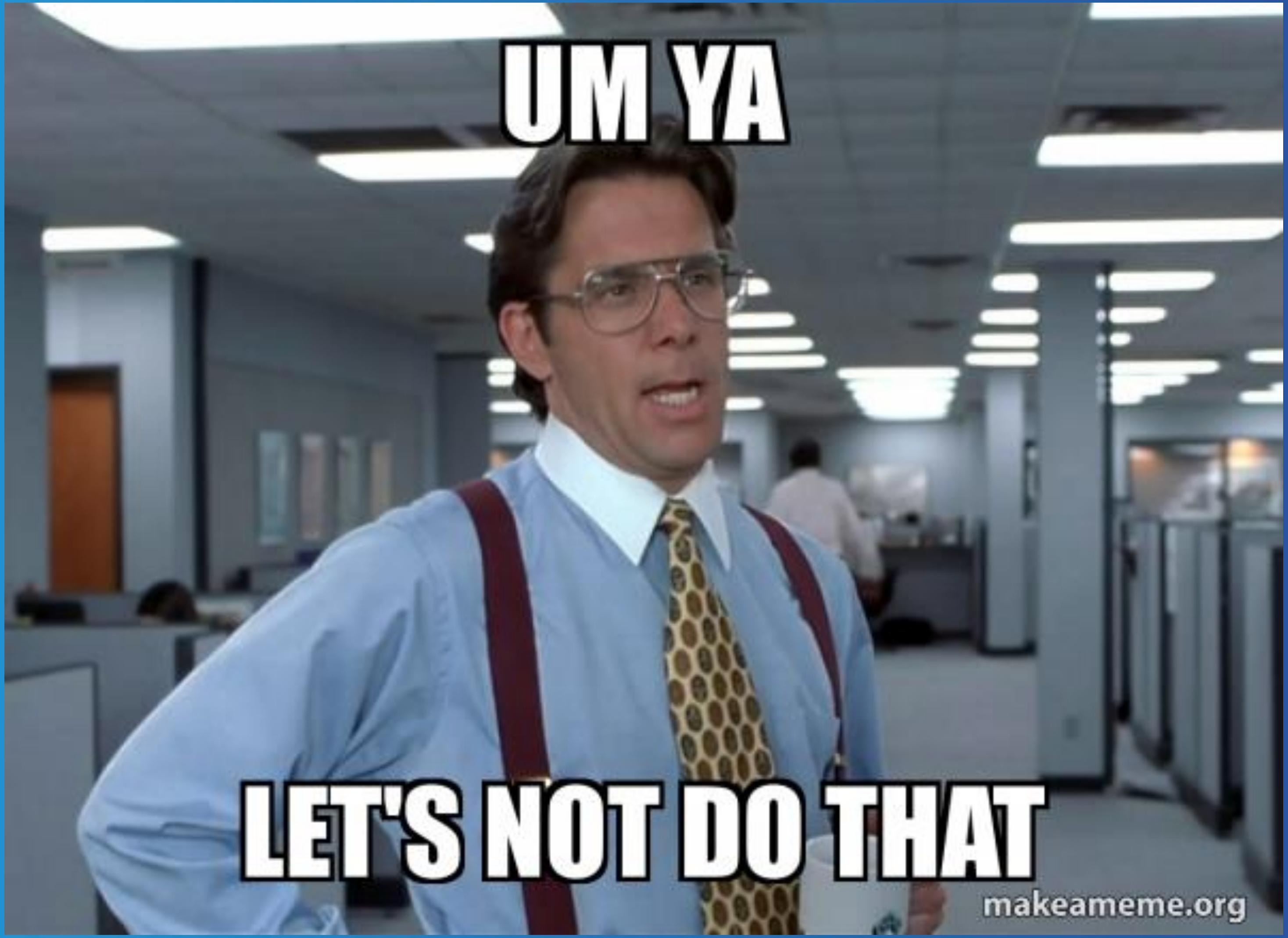
From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.



$$\mathsf{Y} = \lambda f.\ (\lambda x.\ f\ (x\ x))\ (\lambda x.\ f\ (x\ x))$$

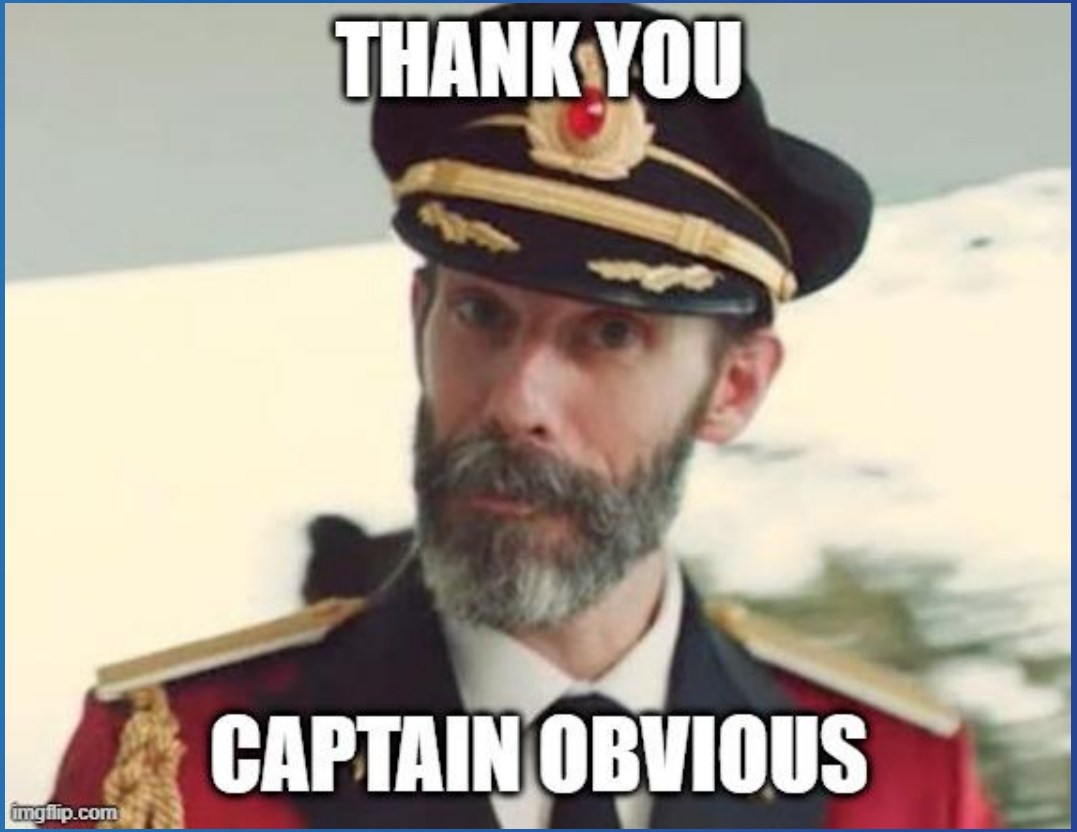
A programming technique that combines the flexibility and power of abstract mathematics with the intuitive clarity of abstract mathematics.

>functional programming



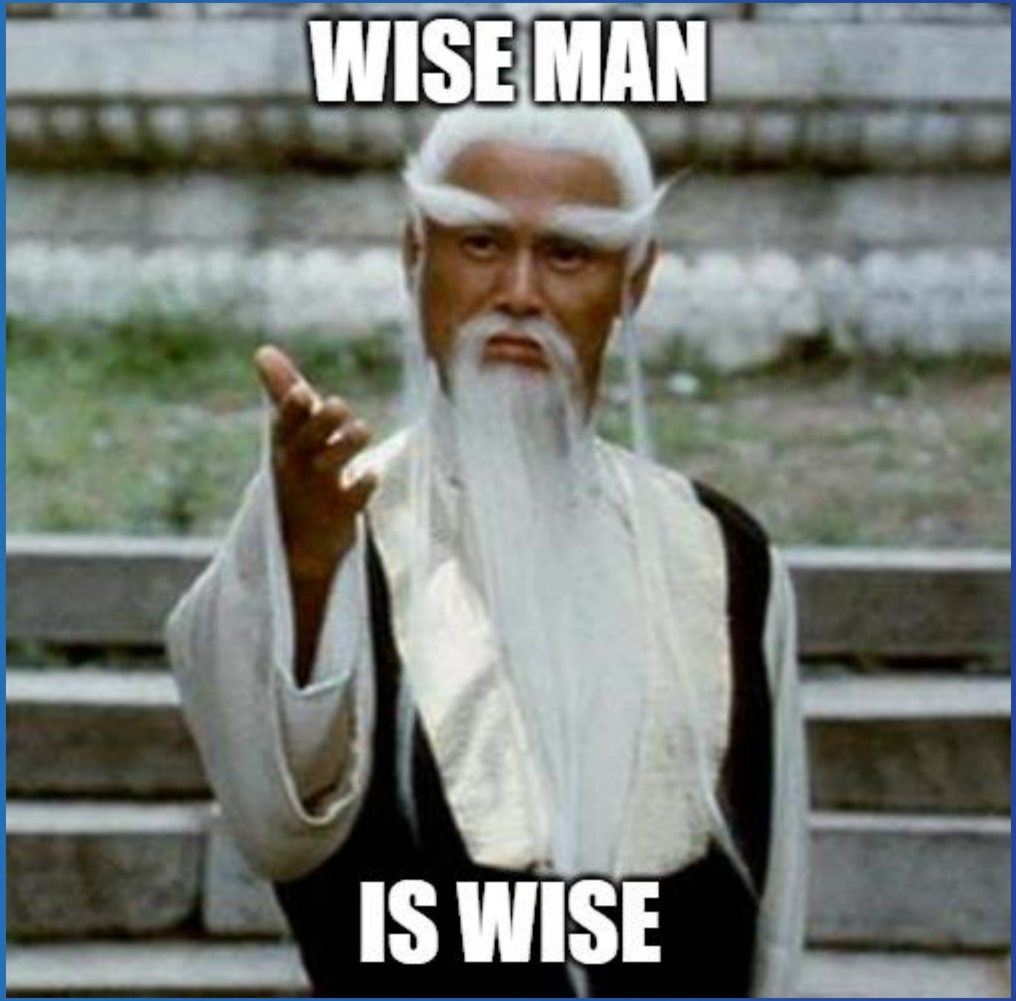
>functional programming

Functional programming is programming that uses functions!



>functional programming

- Use functions as first-class citizens
- Prefer immutable values
- Prefer pure functions

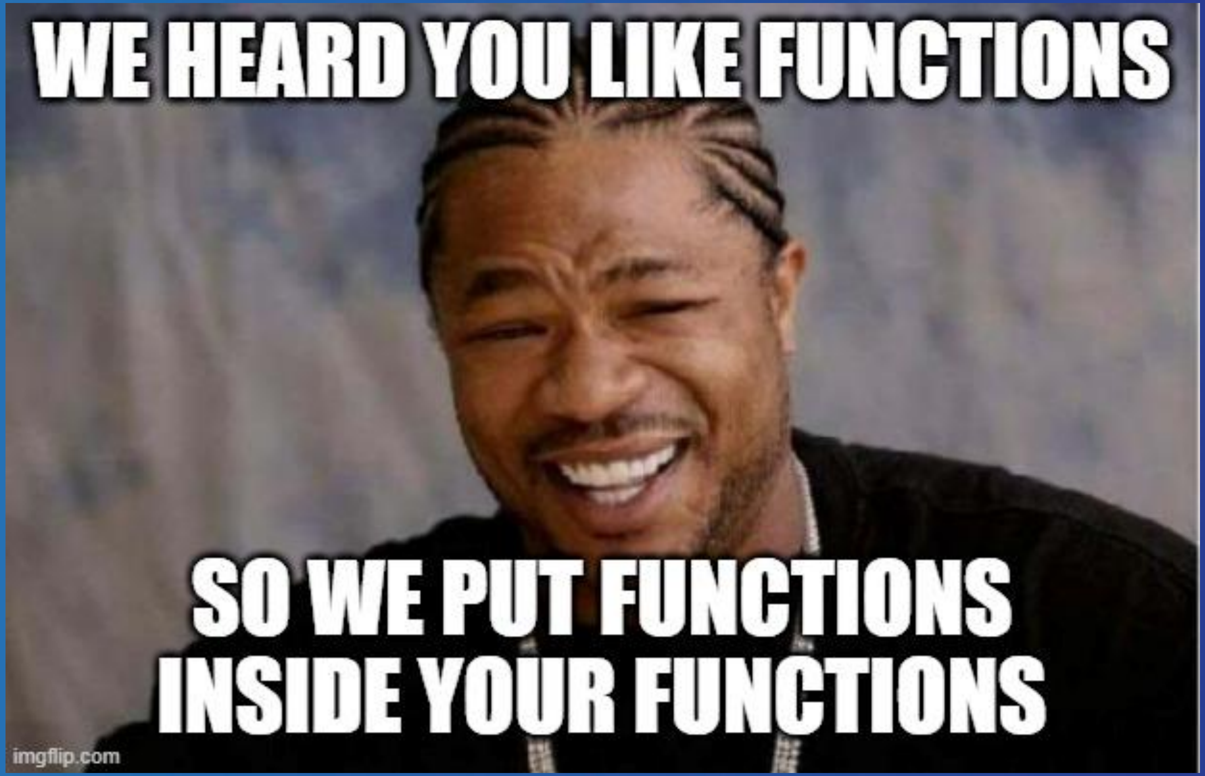


>first-class functions

- Functions **are** values
- Can be assigned
- Can be parameters
- Can be return values
- Can be called

>higher order functions

- E.g. **combine** is a function that takes two functions and returns a function that calls the parameter functions in succession.
- map a.k.a. Select
- filter a.k.a. Where
- reduce a.k.a. Aggregate


WE HEARD YOU LIKE FUNCTIONS
SO WE PUT FUNCTIONS INSIDE YOUR FUNCTIONS
imgflip.com

>object immutability

- Don’t change values
- Simpler state management
- Predictability
- Testability
- Debugging experience

>functional purity

- Don’t touch what was not given to you
- Don’t use globals
- Testability
- Memoization

Enough chitchat,
show us the codes