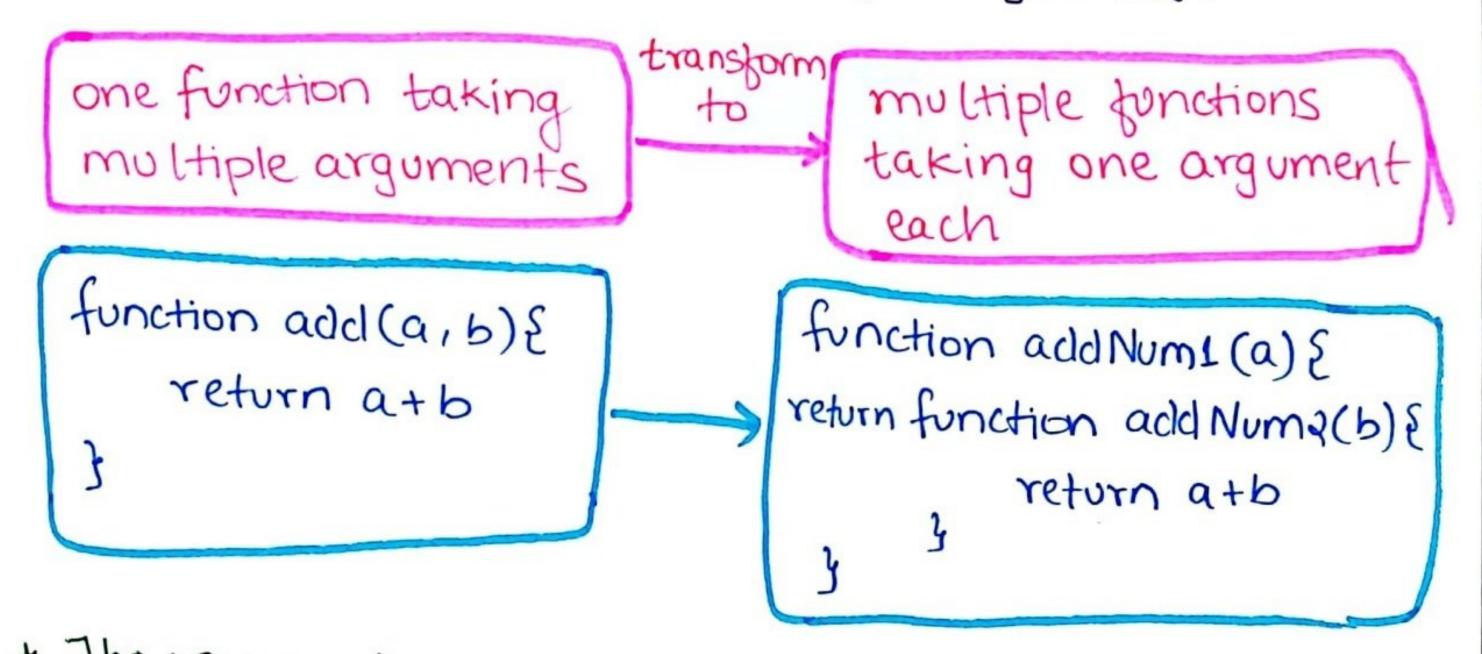
Currying

@cocleWithSimran

Currying is a mechanism where we can translate the evaluation of a function that takes multiple arguments into evaluating a sequence of functions that take a single argument.



function additions has access to variable a due to closures



So now, how do we use this function?

Decause add Number only takes one argument and it returns a function so result of add Numbers of the second argument to add Numbers

Alright, why would someone even do this in the first place??

het's say we just call

@codeWithSimran

addNumi \$ (5)

this will return us a function, so let's store it for tuture use

- -> const addToNum5 = addNum1(5)
- 11 sometime in ture
- -> addToNum5(3)

@codeWithSiman

-> add To Nums (4)

we're trying to run less code in future by storing add Numto add ToNums for future and not running it everytime



Compose

* The clata processing that we do should be obvious

Example.

Ker's say we want to make a paste from 5 ingredients mixed together.

items add, itema blend, new Item add, items

Final Product _ blend

So final product is made from items, items, items and the order dosen't matter

What compose say's is we should be able to do

item 2 add > item 3 blend > new Item add > item 1

Final Product blend

A highly composable environment components can be assembled in various combinations and still get the right output

@ code With Simman





Code example: -

@code WithSimman

Let's say we have a negative number, we want to multiply it with another number and return the absolute value

-2 * 3 -- 6 absolute 6

Let's compose these together

1) const composed Result =

we need to define our own compose func.

const compose = (f,g) >

- 2) const compose = function (func A, funs) &
 - 3) return function (data) {
- return funcB(funcA(data))

>> composed Result (-2) funca (funca (data))

Let's assume multiply By3 and return Absolute are defined.

on line one we're calling compose function that takes a functions as arguments and returns a function (line 3), this function is shored in composed Result. We can now coul composed Result with any data (-2) and if applies func B on result of function

codeWithSimran

