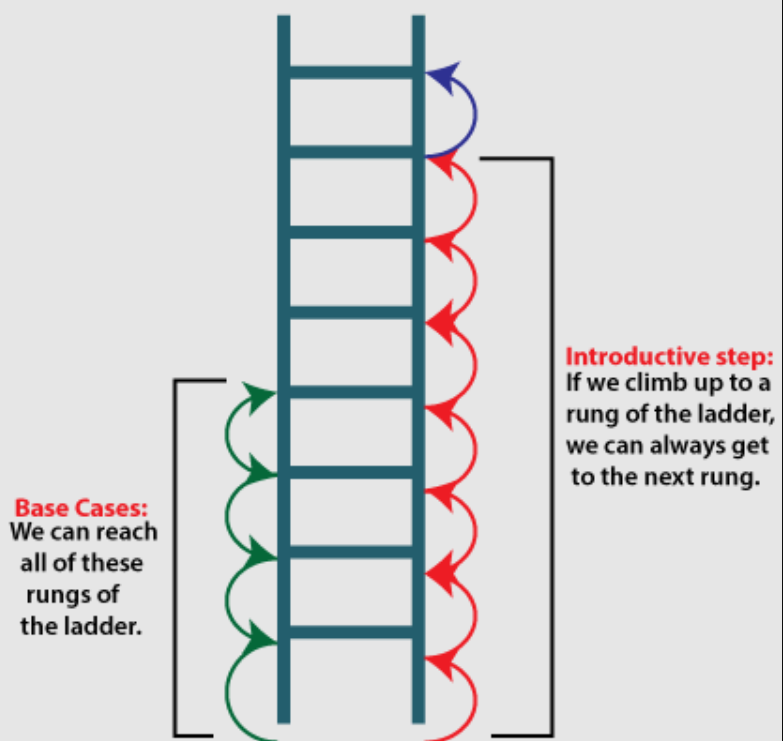
RECURSIVE

**Pseudocode** is a high-level description of the problem you’re trying to solve, in code. It’s written like code, but it’s meant to be closer to human speech.

Recursion is where a function calls itself.



**Base case and recursive case**

*function* *countdown*(i) {

*console*.*log*(i)

*if* (i *<=* 0) { *// base case*

*return*;

    }

*else* { *countdown*(i *-* 1) } *// recursive case*

}

*countdown*(3)

**The call stack with recursion**

Recursive functions use the call stack too! Let’s look at this in action

with the factorial function. **factorial(5)** is written as 5!, and it’s

deined like this: 5! = 5 \* 4 \* 3 \* 2 \* 1. Similarly, factorial(3) is

3 \* 2 \* 1. Here’s a recursive function to calculate the factorial of a

number:

*function* *fact*(x){

*if* (x*===*1){

*return* 1;

    }

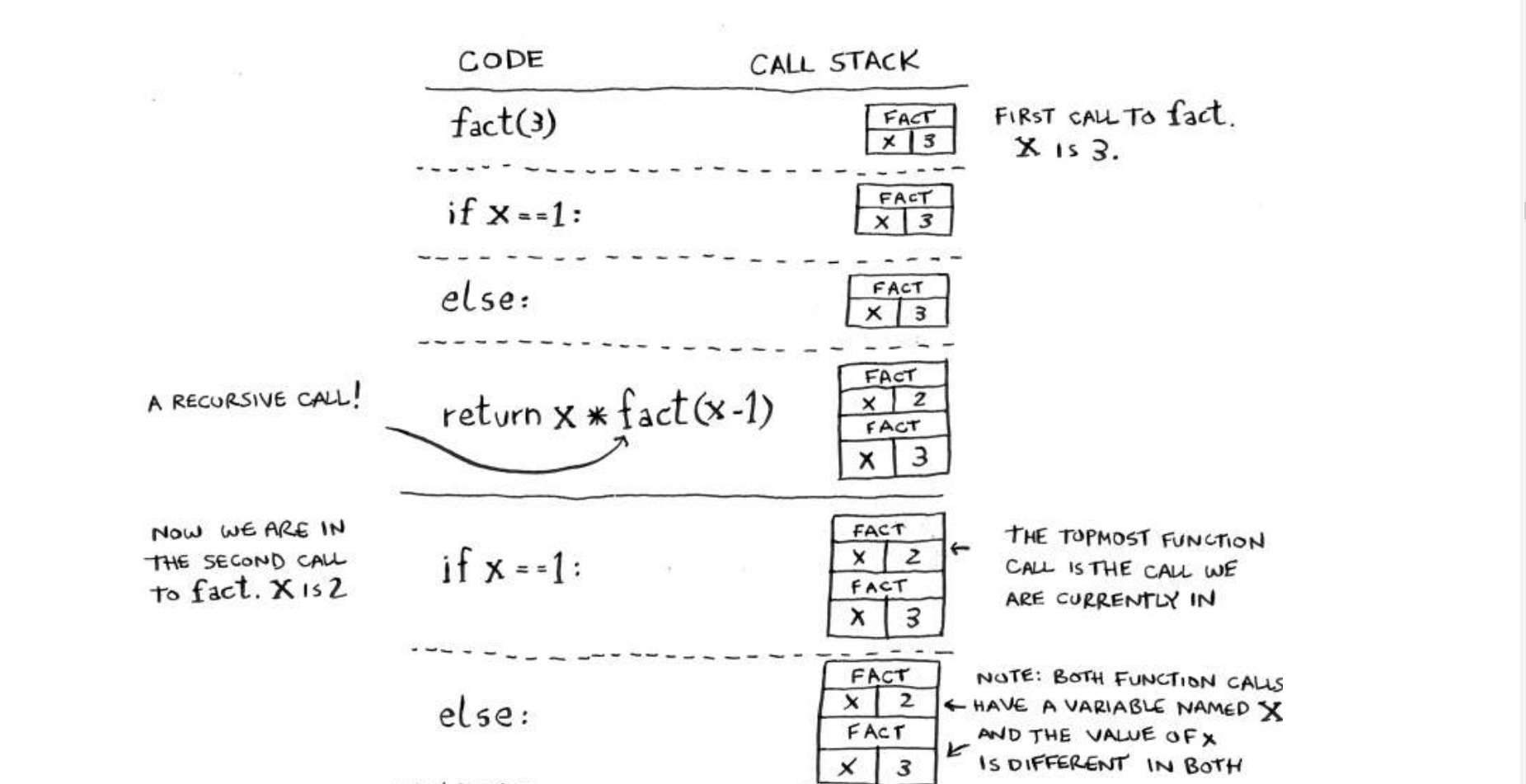
*else* {

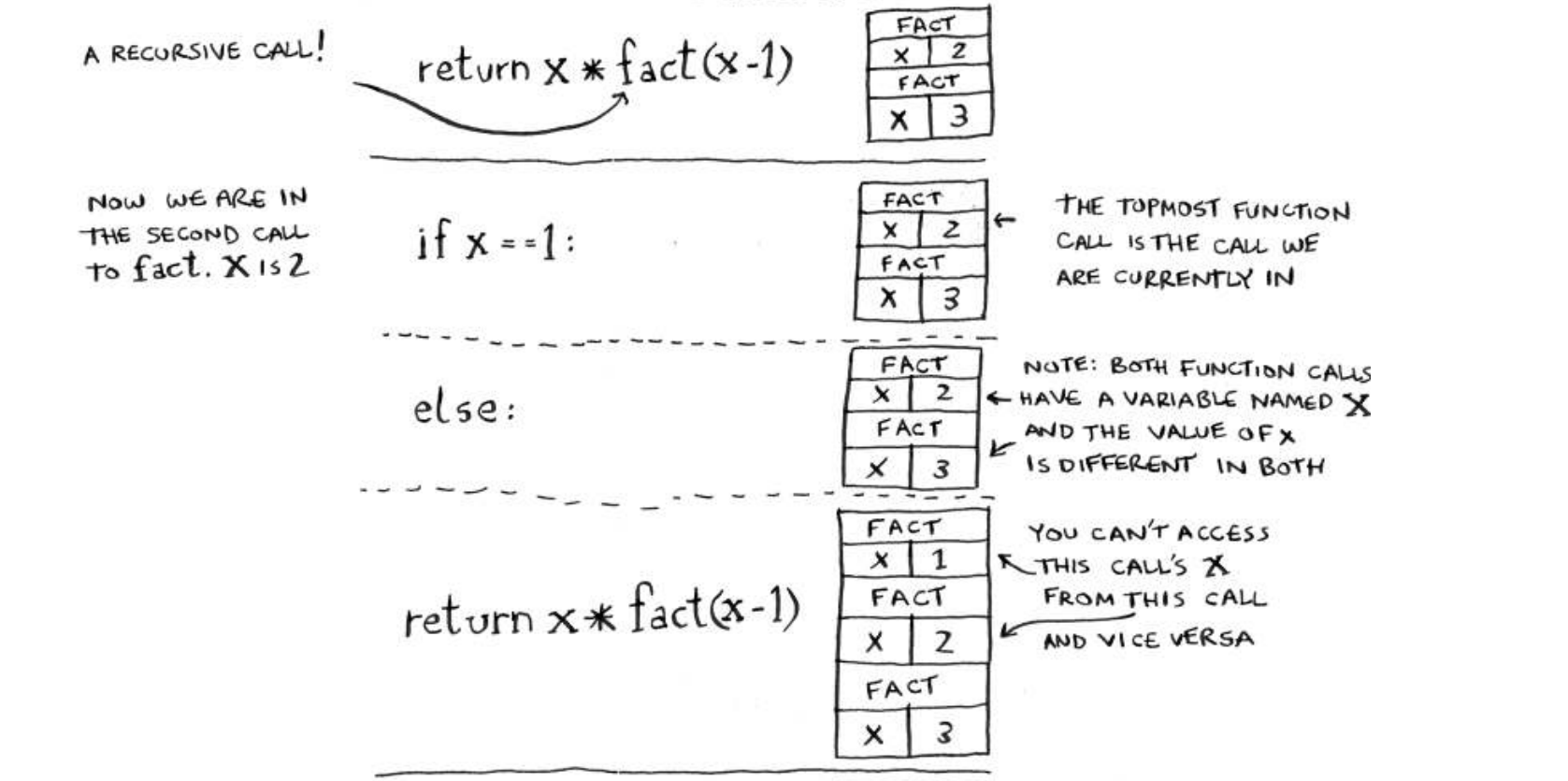
*return*   x *\** *fact*(x*-*1) // a recursive call

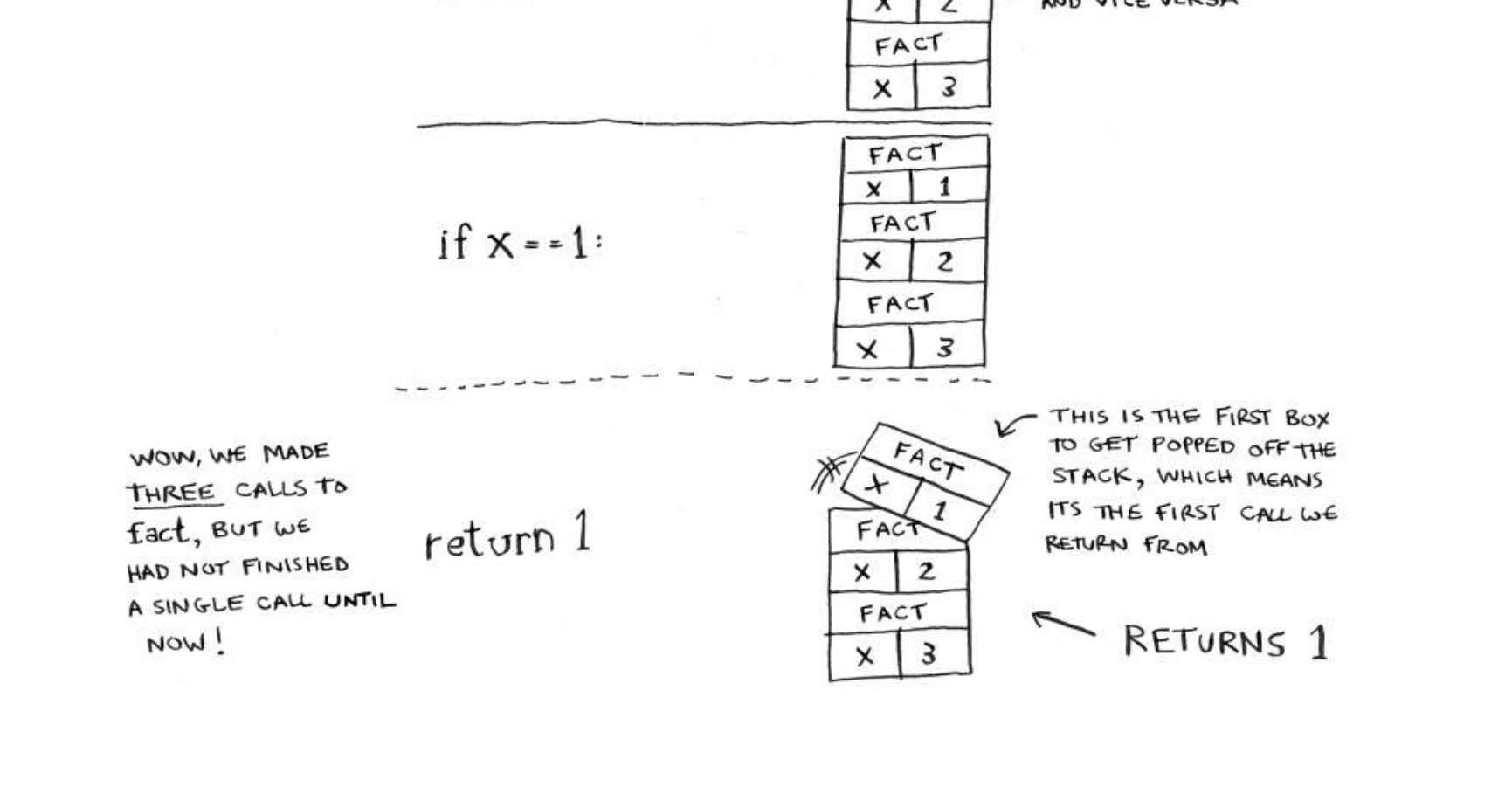
    }

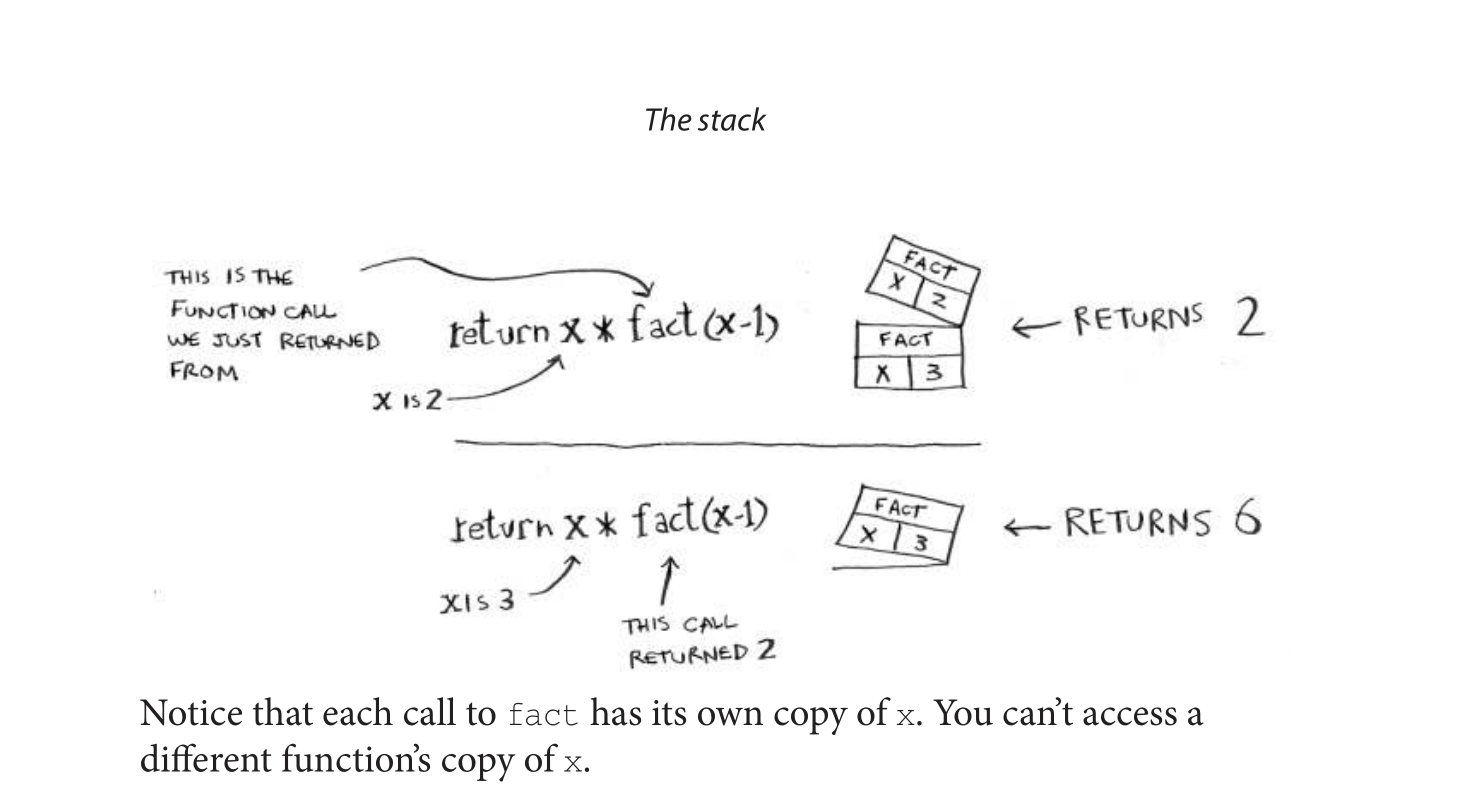
}

*console*.*log*(*fact*(3))









**Recap**

• Recursion is when a function calls itself.

• Every recursive function has two cases: the base case

and the recursive case.

• A stack has two operations: push and pop.

• All function calls go onto the call stack.

• the call stack can get very large, which takes up a lot of memory.