Codesmith

CSPrep

Day 5

Overview

POD

What is recursion?

- What is recursion?
    - What is recursion?

Why is recursion useful?

Imperative vs. declarative programming

The call stack

The base case

Pairing!

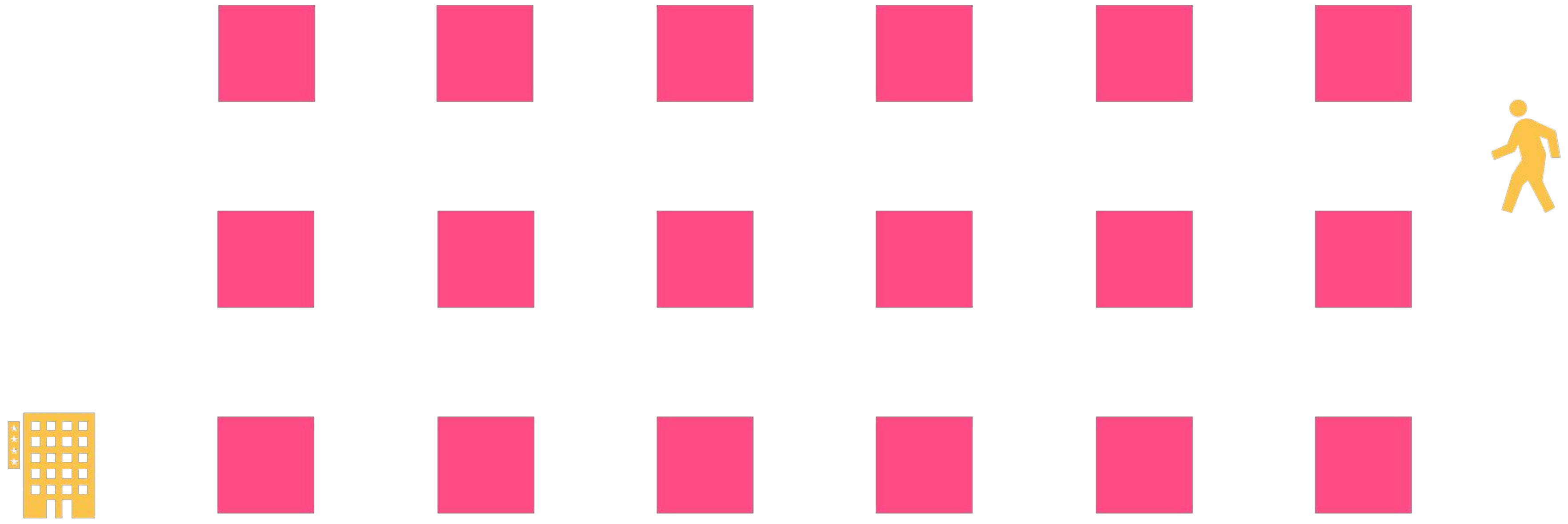
What is recursion?

A method of solving a problem where the solution depends on solutions to smaller instances of the same problem.

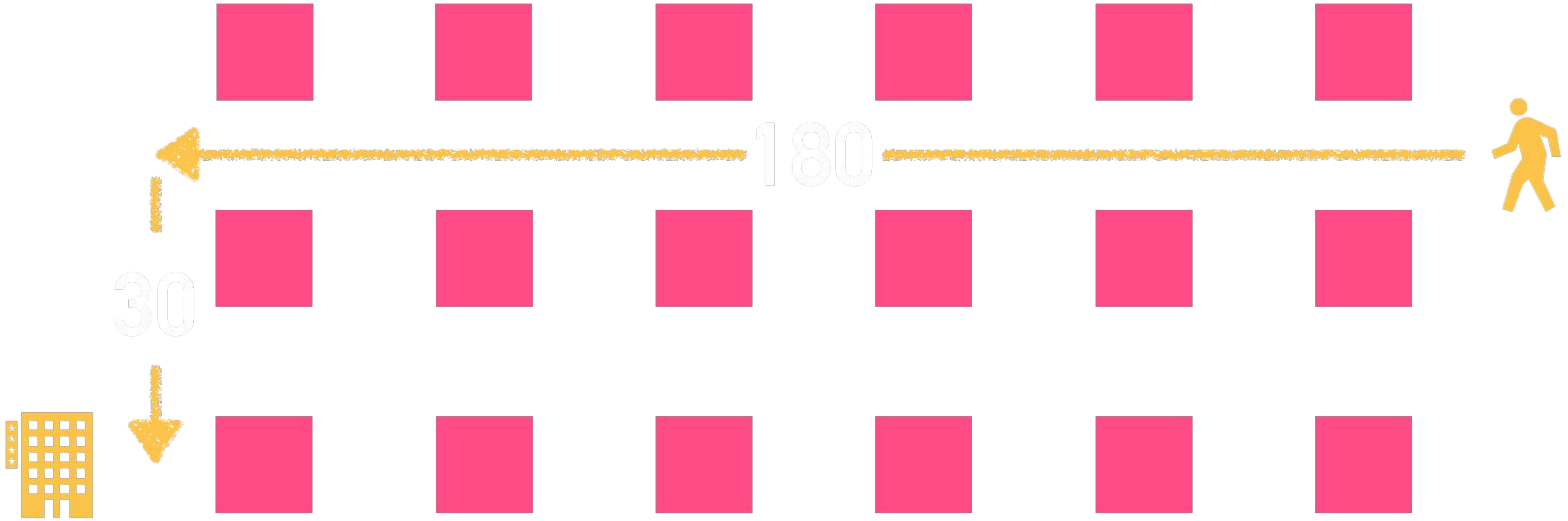
A function that calls itself within its own definition.

A tool for writing declarative programs.

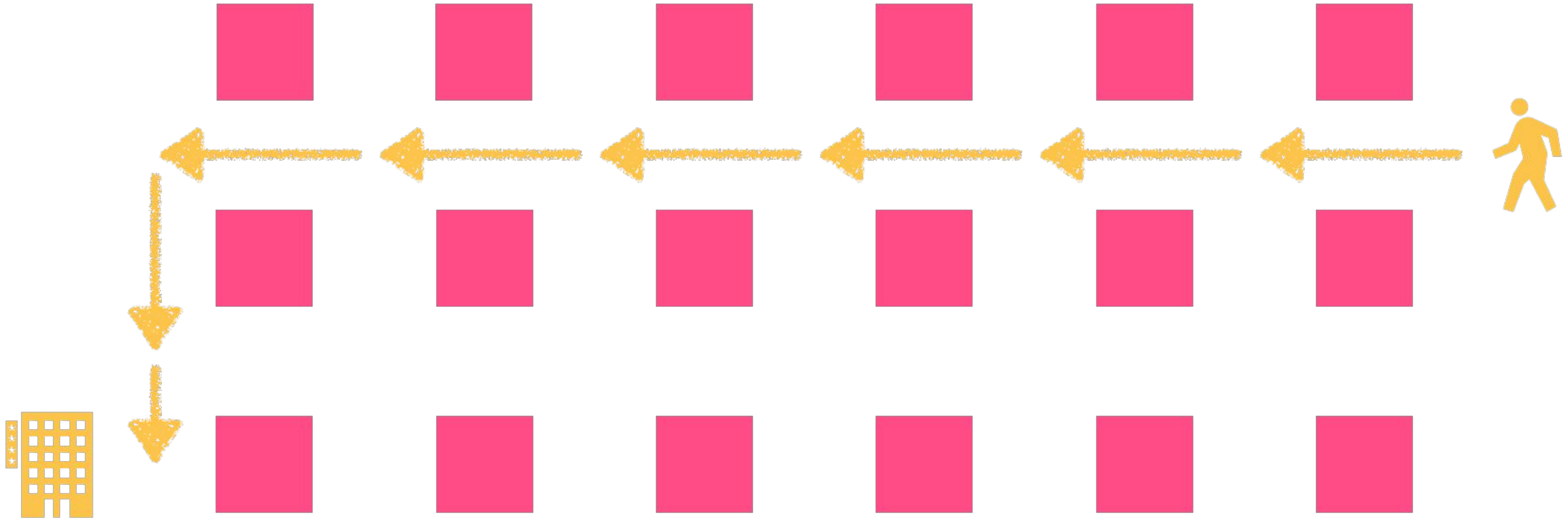
When you walk from one side of town to the other side of town...

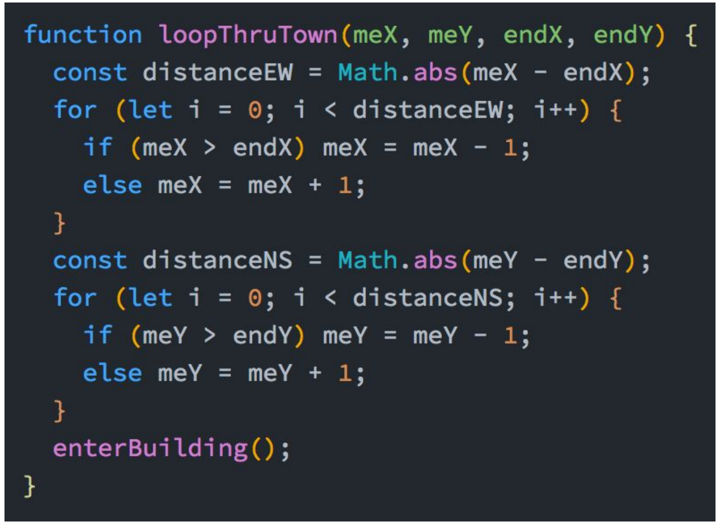


...do you count how many steps you take to know when to take each turn?



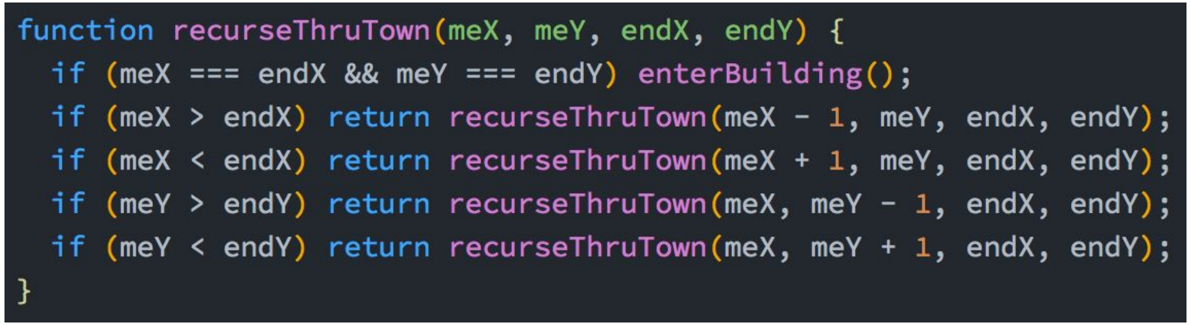
Or do you walk corner to corner, deciding what to do next at each corner?





```javascript
function loopThruTown(meX, meY, endX, endY) {
  const distanceEW = Math.abs(meX - endX);
  for (let i = 0; i < distanceEW; i++) {
    if (meX > endX) meX = meX - 1;
    else meX = meX + 1;
  }
  const distanceNS = Math.abs(meY - endY);
  for (let i = 0; i < distanceNS; i++) {
    if (meY > endY) meY = meY - 1;
    else meY = meY + 1;
  }
  enterBuilding();
}
```

Vs.



```javascript
function recurseThruTown(meX, meY, endX, endY) {
  if (meX === endX && meY === endY) enterBuilding();
  if (meX > endX) return recurseThruTown(meX - 1, meY, endX, endY);
  if (meX < endX) return recurseThruTown(meX + 1, meY, endX, endY);
  if (meY > endY) return recurseThruTown(meX, meY - 1, endX, endY);
  if (meY < endY) return recurseThruTown(meX, meY + 1, endX, endY);
}
```

Why learn recursion?

Why is this important?

Intuitive: We already think recursively in everyday life

Declarative programming

Reduce complexity of iterative code

Elegant way to traverse data structures

The call stack

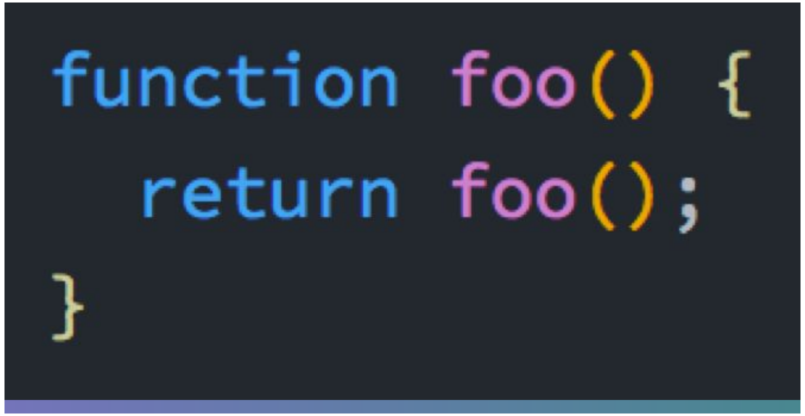
The Call Stack

A finite resource stack implementation that enables and tracks the execution of JavaScript.

As functions are called, they're pushed onto the stack. This new execution environment is called a stack frame.

When functions hit a return keyword, the value following the return is returned to the previous stack frame. The frame is popped from the call stack.

Hmm... what happens here?



```
function foo() {
   return foo();
}
```

The Base Case

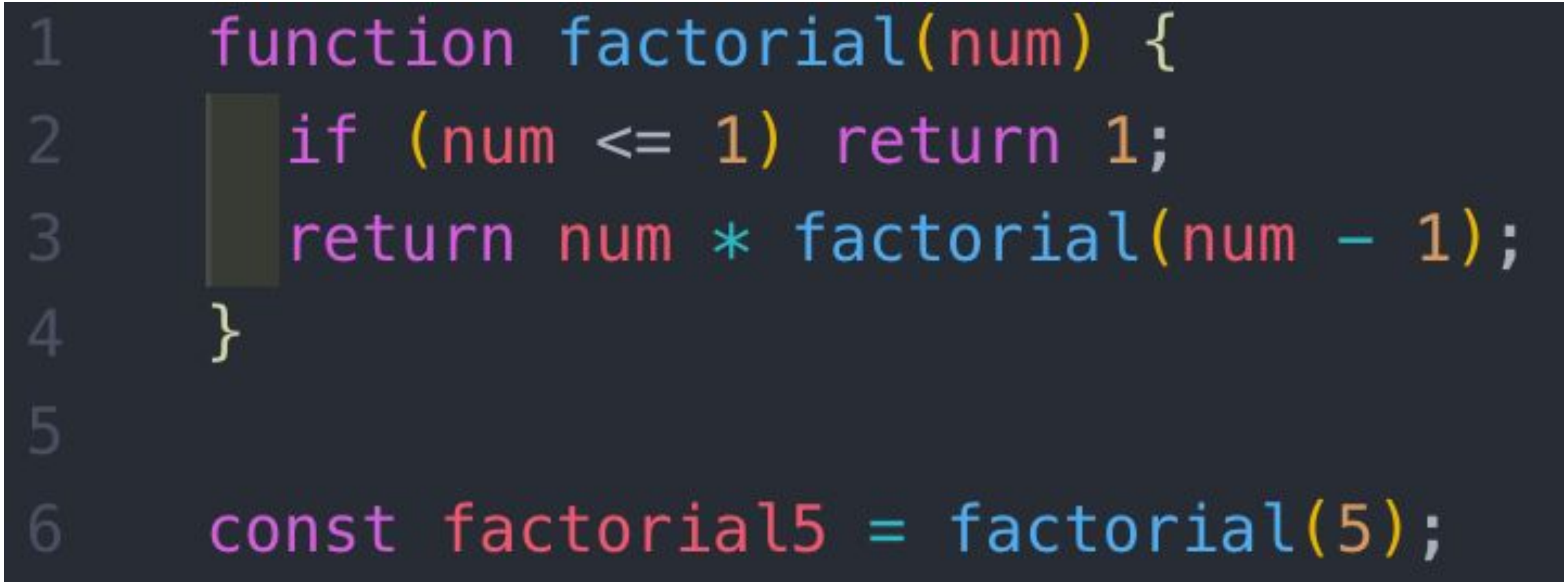
Base Case

The base case returns a value without making any subsequent recursive calls.

Each recursive call must bring the program closer to reaching the base case. If not, stack overflow!

Let's look at some examples

Let’s diagram!



```
function factorial(num) {
    if (num <= 1) return 1;
    return num * factorial(num - 1);
}

const factorial5 = factorial(5);
```

Where do we see recursion?

Sorting

- mergesort, quicksort

Binary search tree traversals

- Calculate height, find a value, add a value, etc.

Graph Traversals

- Depth First Search (DFS)

Combinations and Permutations

Review

Non-recursive functions tend to describe how to get to a solution. Recursive solutions describe what the solution is.

Writing recursive functions often forces you to write declarative code instead of imperative code.

JavaScript runtimes use a call stack to track the execution of a program, pushing function calls onto that stack and popping when a return statement is met.

All recursive functions must approach a non-recursive base case.

Let’s pair!