Recursion

3 min

So, what is recursion?

Recursion is a computational approach where a function calls itself from within its body. Programmers use recursion when they need to perform a similar action multiple times in a row until it reaches a predefined stopping point, also known as a base case.

Let's think about this in the context of our factorial example. Below is the beginning of a recursive implementation of factorial. This code is all in **index.js**, to the right.

```
const recursiveFactorial = (n) => {
  if (condition){
    console.log(`Execution context: ${n}`);

    recursiveFactorial(n - 1);
  }
};
```

Within the recursiveFactorial() function, we want to check whether a condition is met. If it is, then we print the value of n and return a call to recursiveFactorial(n - 1).

Can you think of a condition that will result in the following response when we call recursiveFactorial(4)?

```
Execution context: 4
Execution context: 3
Execution context: 2
Execution context: 1
```

The correct answer is n > 0. At this point, we have the beginnings of a recursive function, but we're still not returning anything.

Instructions

1. Checkpoint 1 Passed

1.

Change the condition in the if statement to something that will prevent recursiveFactorial() from calling itself if n is less than 1.

Hint

Change /*SOME CONDITION*/ to n > 0.

index.js

```
const recursiveFactorial = (n) => {

  if (n > 0) {

    console.log(`Execution context: ${n}`);
    recursiveFactorial(n - 1);
  }
}

const recursiveSolution = recursiveFactorial(4);
console.log(recursiveSolution);

module.exports = {
  recursiveFactorial
};
```