Recursive Case

3 min

In the last exercise, you created a condition (n > 0 or n >= 1). This condition is important, because it defines whether or not recursiveFactorial() calls itself. We call this if block the *recursive case*.

In recursion, the *recursive case* is the condition under which a function calls itself. We call this the recursive case because, as mentioned last exercise, recursion is defined as a process when a function calls itself.

At the end of last exercise, your output should have looked like:

Execution context: 4
Execution context: 3
Execution context: 2
Execution context: 1
undefined

At this point, there are a couple of shortcomings in the implementation that are worth mentioning:

- Calculating the product of the numbers – while we do access all of the numbers that need to be multiplied, we do not calculate their product.
- recursiveSolution is set to undefined – the value set to recursiveSolution (see **index.js** to the right) is undefined, because we never returned anything from recursiveFactorial().

Instructions

1. Checkpoint 1 Passed

1.

In your function, return the product of n and your call to recursiveFactorial().

After you run your code, you should see that the value saved to recursiveSolution has changed. Is it what you expect?

Hint

You need to return the product of n and your call to recursiveFactorial(n - 1).

You can do this with the following syntax:

return n * recursiveFactorial(n - 1);

index.js

```
const recursiveFactorial = (n) => {
  if (n > 0){
    console.log(`Execution context: ${n}`);

    return n * recursiveFactorial(n - 1);
```

```
  }
}

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

module.exports = {
  recursiveFactorial
};
```