Base Case

3 min

The solution to the last exercise resulted in the following output:

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

Notice, the value saved to recursiveSolution changed from undefined to NaN (not a number).

Why is recursiveSolution not a number? The short answer: we didn't define a *base case*. To understand the need for a base case, it's worth discussing the call stack that JavaScript creates when you call recursiveFactorial().

If you were to call:

```
recursiveSolution = recursiveFactorial(3)
```

JavaScript would create a call stack with the following events:

1. recursiveFactorial(3) = 3 * recursiveFactorial(2)
2. recursiveFactorial(2) = 2 * recursiveFactorial(1)
3. recursiveFactorial(1) = 1 * recursiveFactorial(0)

The return value associated with each function call depends on the value returned by the n - 1 context. Because the current implementation does not return a number for recursiveFactorial(0), the result of (3) is NaN. This leads to an NaN solution for each of the contexts above it.

We need a *base case* to address the NaN returned from the n === 0 context. The factorial function should return a number when n === 0.

Instructions

1. Checkpoint 1 Passed

1.

We set recursiveSolution equal to the value returned from recursiveFactorial() with 0 as the argument.

Run the code. You should see undefined in the terminal.

Hint

Change the line:

```
const recursiveSolution = recursiveFactorial(4);
```

to:

```
const recursiveSolution = recursiveFactorial(0);
```

2. Checkpoint 2 Passed

2.

Inside recursiveFactorial(), add an if statement that returns 1 when n is equal to 0.

Hint

In the code below, we create an if statement that returns 1

```
const recursiveFactorial = (n) => {
  // Add a condition below
  if (/*SOME CONDITION*/) {
    return 1;
  }

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

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

Set the condition in the example above to n == 0.

3. Checkpoint 3 Passed

3.

Set recursiveSolution equal to the value returned from recursiveFactorial() with 5 as the argument.

Hint

Change the line:

```
const recursiveSolution = recursiveFactorial(0);
```

to:

```
const recursiveSolution = recursiveFactorial(5);
```

index.js

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

  // Add a condition below

  if (n === 0) {

    return 1

  }


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

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

const recursiveSolution = recursiveFactorial(5);
console.log(recursiveSolution);

module.exports = {
  recursiveFactorial
};
```