**Recursion**

*Definition*: A function that calls itself

*Procedure:* Invoke the same function with a different input until you reach your base case (the condition when the recursion ends)

*Example:*

function countDown(num){

if(num <= 0) {

console.log("All done!");

return;

}

console.log(num);

num--;

countDown(num);

}

countDown(3)

// print 3

// countdown(2)

// print 2

// countdown(1)

// print 1

// countdown(0)

// print “All Done”

**Helper Method Recursion**

*Definition:* A function calls itself inside another function

*Sample Code:*

function outer(input) {

var outerScopedVariable = [];

function helper(helperInput) {

helper(helperInput--);

}

helper(input);

return outerScopedVariable;

}

*Example:*

function collectOddValues(arr){

let result = []; // this array used to put the odd number in

function helper(helperInput){

if(helperInput.length === 0) {

return;

}

if(helperInput[0] % 2 !== 0){

result.push(helperInput[0])

}

helper(helperInput.slice(1))

}

helper(arr);

return result;

}

collectOddValues([1,2,3,4,5,6,7,8,9])

**Pure Recursion**

For arrays, use methods like slice, the spread operator, and concat that makes copies of array so you do not mutate them.

Remember that strings are immutable so you will need to use methods like slice, substr, or substring to make copies of strings

To make copies of Object use Object.assign, or the spread operator