

# 1.- AN INTRODUCTION TO FUNCTIONS, EXECUTION CONTEXT AND THE CALL STACK

- We execute the code line by line

What happens when javascript executes (runs) my code?

```
const num = 3;  
function multiplyBy2 (inputNumber){  
  const result = inputNumber*2;  
  return result;  
}  
const name = "Will"
```

As soon as we start running our code, we create a **global execution context**!

Which:

- ♥ Thread of execution (Is parsing and executing the code line by line)

- ♥ Live memory of variables with data (Known as a Global Variable Environment)

## Running/ Calling/ Invoking a Function

When you execute a function, you create a new execution context comprising:

- The tread of execution (We go trough the code **in the function** line by line one more time)
- A local memory (**Variable environment**) where anything defined in the function is stored

## 2.- HOW TO UNDERSTAND CALLBACKS & HIGHER ORDER FUNCTIONS

**CALLBACKS & HIGHER ORDER FUNCTIONS**

Call stack

global()

output = multiplyBy2(4)

Local Memory

inputNumber: 4

result: 8

num++

function tenSquared() {  
  return 10\*10;  
}

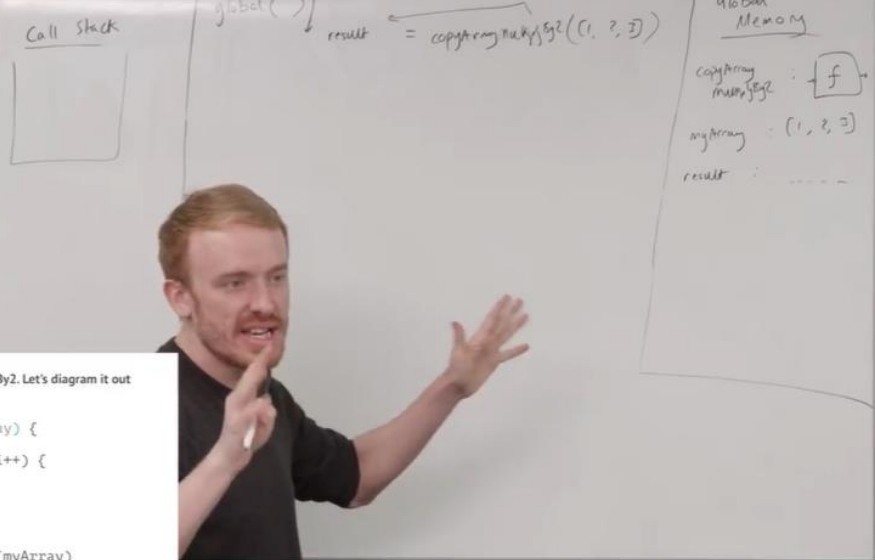
Create a function 10 squared

Takes no input

Returns 10\*10

How do we do it?

## CALLBACKS & HIGHER ORDER FUNCTIONS



Now suppose we have a function copyArrayAndMultiplyBy2. Let's diagram it out

```
function copyArrayAndMultiplyBy2(array) {
  let output = [];
  for (let i = 0; i < array.length; i++) {
    output.push(array[i] * 2);
  }
  return output;
}
const myArray = [1, 2, 3]
let result = copyArrayAndMultiplyBy2(myArray)
```



- ♥ EVERYTHING IS STORED IN THE MEMORY AND EXECUTION
- ♥ BENDING OUT TOGETHER!

## CALLBACKS & HIGHER ORDER FUNCTIONS

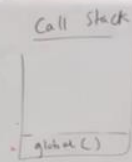


Now suppose we have a function copyArrayAndMultiplyBy2. Let's diagram it out

```
function copyArrayAndMultiplyBy2(array) {
  let output = [];
  for (let i = 0; i < array.length; i++) {
    output.push(array[i] * 2);
  }
  return output;
}
const myArray = [1, 2, 3]
let result = copyArrayAndMultiplyBy2(myArray)
```

## CALLBACKS & HIGHER ORDER FUNCTIONS

Call Stack



global



result = copyArrayAndDivideBy2([1, 2, 3])

array [1, 2, 3]  
12  
output [0.5, 1, 1.5]

Local Memory  
array : [1, 2, 3]  
output : [0.5, 1, 1.5]

return

result

Global Memory

copyArrayAndDivideBy2 : f  
myArray : [1, 2, 3]  
result : [0.5, 1, 1.5]

What if want to copy array and divide by 2?

```
function copyArrayAndDivideBy2(array) {  
  let output = [];  
  for (let i = 0; i < array.length; i++) {  
    output.push(array[i] / 2);  
  }  
  return output;  
}  
const myArray = [1, 2, 3]  
let result = copyArrayAndDivideBy2(myArray);
```



# THE END!