**Closures in Javascript**

**// There is an inner function inside an outer function**

**// Summary**

**// Variables in scope of outer function are stored in the closure of inner functions, such that they do not cease to exist when the inner function finishes executing**

A nice way to understand this is imagine a counter function that counts how many times the function is called

First Understand the scoping in Jaavascript

There is

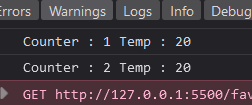
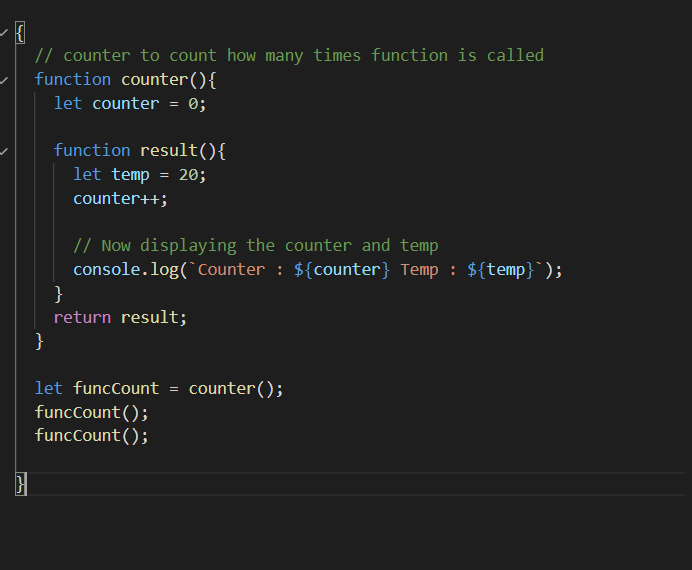
1. Global scoping
2. Block level scoping
3. Function scoping

According to lexical scoping, the scopes can be nested and the inner function can access the variables declared in its outer scope. For example:



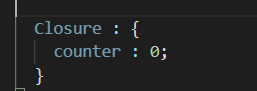
here sayHI() function can access the variable “message” present inside the greeting() function

Seek this to understand closures in deep : <https://medium.com/@prashantramnyc/javascript-closures-simplified-d0d23fa06ba4>



Understand what is happing above

1. We store data returned by counter function inside the funcCount variable
2. We see counter actually returns another function // In javascript functions are first class citizens, so they can be passed as arguments, returned by another function and stored in a variable
3. Also when counter function is called first of all counter variable is created and value is aligned to 0, and then another function result is returned by counter function
4. Now counter function is finished executing
5. So now every variables created inside counter function’s scope will cease to exist
6. So counter variable also cease to exist
7. Now funcCount actually has result function body stored in it
8. So when we call funcCount() // result() function gets executed
9. And above result is seen
10. It makes sense for result() function to display value of temp, since when result() function starts to execute everytime we call it, temp variable is created and value is stored in it and hence displayed
11. But why result() function is also displaying the value of counter
12. This is due to something called closures in javascript
13. When there is another function inside a function, the inner function actually stores the variables defined in it’s parent scope inside it’s closure. Think closure as some kind of object present in inner function that stores the variables of it’s parent’s function’t scope.



1. So when the counter function was executed very first time, result() function actually store the counter in it’s counter.
2. So everytime result() function was called it was able to perform the operation in the counter
3. But temp is created everytime result() starts to execute and cease to exist when result() finishes executing, but counter is stored in it’s closure so when counter() finishes executing, counter doesn’t cease to exist and when result() is called again next time, result() operates on the latest value of result