UNDERSTANDING CLOSURES

A closure is an inner function that has access to the outer (enclosing) function's variables—scope chain. The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function's variables, and it has access to the global variables.

The inner function has access not only to the outer function's variables, but also to the outer function's parameters. Note that the inner function cannot call the outer function's *arguments*object, however, even though it can call the outer function's parameters directly.

LET'S UNDERSTAND CLOSURE WITH AN EXAMPLE STEP BY STEP

```
function greet(whattosay){
  return function(name){
    console.log(whattosay + " " + name)
  }
}
var sayHi=greet("Hi");
console.log(sayHi)
sayHi("siddharth")
```

STEP 1

When the code starts we have a global execution context.

When JS engine hit the Line `var sayHi=greet("Hi")` . A new global execution context will create for this function and it take the argument 'Hi' from this function and store the value of '`whattosay`' in memory space .after that it will returns a new function object.

STEP 2

After returning the function, object execution context for greet will be disappeared. **But memory space for this context will not be destroy.**

STEP 3

In step 3 JS engine will call `sayHi("siddharth")` function . A new global execution context will create for this function and it take the argument 'siddharth' from this function and store the value of 'name' in memory space .

STEP 4

After step 3 JS engine will go to '`console.log(whattosay + " " + name)`' this line. JS engine will find `whattosay` variable and will search for the value of `whattosay` in scope chain .first it will go its own lexical scope and it will not found any value for the `whattosay` then it will go to outer lexical scope where reference for `whattosay` memory space already exist because in step two only execution context disappeared not memory space.

BY THIS FOUR STEPS INNER FUNCTION HAS ACCESS TO THE OUTER (ENCLOSING) FUNCTION'S VARIABLES