

CLOSURES IN JAVASCRIPT




JS

Hi EVERYONE!

In Today's post, we will see the concept of **closures in Javascript**. Closures are one of the easiest concepts of Javascript.

Don't forget to save this post so you can access it whenever you needed.

JS

Do You know

In many languages, you **can't use or access** the global variables inside the function.

But In Javascript, a function has the **access to all the variables** that have been declared in the **higher scope or outside** the function.

And this feature is called as **Closure**.

Closure in Nested Functions

As we have seen that a function can access the global variables or the variables outside of it.

so in a similar manner suppose if there are two functions say `innerFunction` and `outerFunction` and `innerFunction` is inside the `outerFunction`. So the `innerFunction` will have the access to the variables and parameters of the `outerFunction`.

This is the most common use of closures.

Example:



```
function outerFunction(a) {  
    let b = 10;  
    return function innerFunction(){  
        let sum = a+b;  
        console.log("The sum is of a and b is: " + sum);  
    }  
}  
  
let result = outerFunction(5);  
result();
```

The sum is of a and b is: 15

!!! Attention Please:

In the above example, I made two functions(**outerFunction** and **innerFunction**). **innerFunction** is inside the **outerFunction**. **a is the parameter** and **b is the variable** of the **outerFunction** but i can access it inside the **innerFunction**. This is known as **closures** in nested functions but (see the next slide...)

How the Program is working?

The outerFunction is returning the innerFunction

```
return function innerFunction(){  
  let sum = a+b;  
  console.log("The sum is of a and b is: " + sum);  
}
```

The innerFunction taking the values of a and b from outerFunction

```
function outerFunction(a) {  
  let b = 10;  
  return function innerFunction(){  
    let sum = a+b;  
    console.log("The sum is of a and b is: " + sum);  
  },  
}
```

Then I called the `outerFunction` and give it the argument "5" and stored that call in the variable `result`.

```
let result = outerFunction(5);
```

Actually what happened here is: The `outerFunction` returned or gave me the `innerFunction` and `result` variable holds that `innerFunction`.

and due to this you can say that: `result` variable also becomes the function because it holds the `innerfunction`. Then i called the `result`.

```
let result = outerFunction(5);  
result();
```


So even after the outerFunction stopped executing. (here it stopped its execution):

```
let result = outerFunction(5);
```

We are still getting the output from the inner function.

```
The sum is of a and b is: 15
```

Conclusion:

A closure is a function having access to the parent scope, even after the parent function has closed.