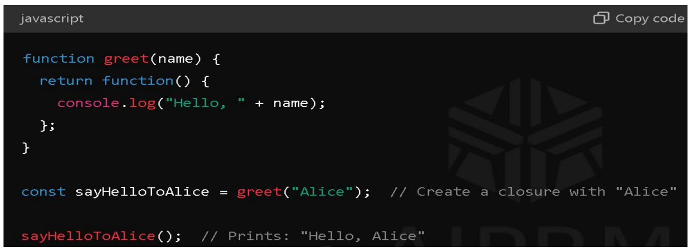
What is a Closure?

A **closure** is like a special tool that lets a function remember the environment it was created in, even after it's been run.

Example:-



```javascript
function greet(name) {
  return function() {
    console.log("Hello, " + name);
  };
}

const sayHelloToAlice = greet("Alice");  // Create a closure with "Alice"

sayHelloToAlice();  // Prints: "Hello, Alice"
```

Explanation:

1. **Outer Function**: greet(name) takes a name and returns an inner function.
2. **Inner Function**: The inner function remembers the name it was given when greet was called.
3. **Closure**: sayHelloToAlice is a closure that remembers the name "Alice" and prints a greeting.

So, the inner function can still use "Alice" even after the greet function has finished.

Why It's Useful:

- **Remember State**: The function counter remembers how many times it has been called.
- **Private Data**: The variable count is hidden from outside but accessible to the inner function.

In short, closures allow functions to keep using variables from their original environment even after they've finished running.

↳ JavaScript is Synchronous.

Date __/__/__
Page ______

Closures

→ "Closure" is a function bind together with its lexical environment.

★ OR

function along with its lexical scope.

Note:- We can assign:—

① function to a variable

e.g.

```
Var a = function y() {
        console.log (a);-
    };
```

② We can pass a function into a function.

e.g.

```
x( function y(){
      Console.log (a);
   });
```

③ Return this function from a function.

e.g.

```
function x(){
 Var a = 7;
  function y() {
     console.log(a);
  } return y;
}
~~return x();~~ var z = x();
```

→ It will return the function itself. & x() will no longer exsist.

→ Not just function was return but a closure is return.

Date ___ Page ___

Console.log (z); ⟶ this will give function y().
↓ i.e.

```
//- - - - - -
Suppose there are
1000 lines. then after
that if we pass
```

```
f y(){
  console.log(a);
}
```

z(); ⟶ o/p:- 7

---

* **Corner Cases:-**

e.g.

```
function x(){
  var a=7;
  function y(){
  console.log(a);
  }
  a=100;
  return y;
}
var z=x();
console.log(z);
//......
z();
```

⟶ o/p:- 100

Date __/__/__
Page ____

e.g. ② :—

```
function z(){
 var b = 900;
 function x(){
   var a=7;
   function y(){
   console.log(a, b);
  }
   y();
 }
 x();
}
z();
```

O/p :— 7
900

→ Here we try to access parent's scope $ parent's parent scope also.

* Uses of closures :—

↳ Module Design Pattern
↳ Currying in JS
↳ Functions like once → {It only runs one)
↳ Memoize
↳ Maintaining state in async world
↳ set Timeouts
↳ Iterators.
and many more . . . .