JavaScript: Lexical Environment, Closures, and Scopes – Notes

1. Lexical Environment

A lexical environment consists of:

- Environment Record: variables in the current scope

- Outer (Parent) Reference: link to the parent scope

This determines how functions access variables from their outer scope.

2. Closure

Closure = Function + Lexical Environment

A closure is created when an inner function remembers variables from its outer function,

even after the outer function has returned.

3. Scope Behavior

var:

- Function-scoped

- Global if declared outside a function

- Does not follow block scope

let / const:

- Block-scoped

- Not added to global object

- Exist in the Temporal Dead Zone (TDZ) before initialization

4. Lexical Environment Chain (Visual)

Global → Outer Function → Inner Function

5. Closure Example

function outer() {

```
let count = 0;
return function inner() {
count++;
console.log(count);
};
}
let x = outer();
x(); // remembers count even after outer returns
```

6. Key Differences

Lexical Environment = static scope structure

Closure = runtime memory behavior using lexical environment