Execution Context

an execution context is an internal concept that helps manage the execution of code. It keeps track of variables, functions, and the scope chain during the runtime of a program. Each time a function is called, a new execution context is created.

Let's break down the execution context into its key components:

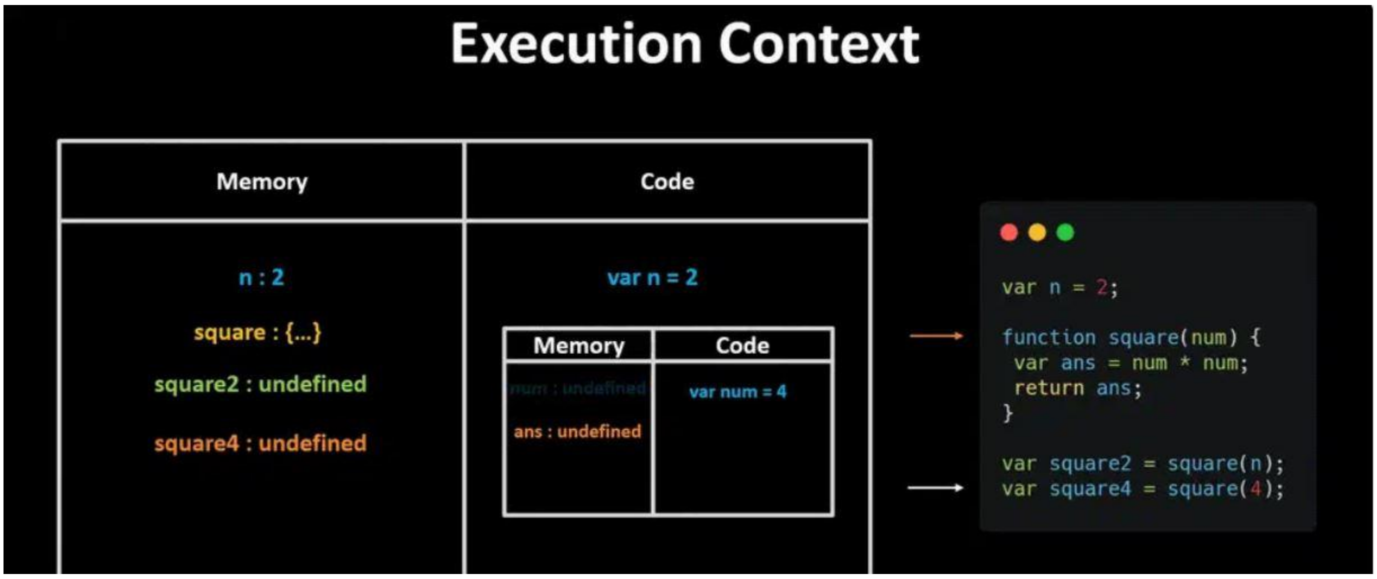
1. Global Execution Context:
- The global execution context is the first execution context created when your JavaScript code starts running.
- It represents the global scope, which includes variables and functions defined outside any function.
- It sets up the global object (`window` object in a browser) and the special variable `this` to refer to the global object.
- The global execution context remains active throughout the entire lifespan of your JavaScript program.

2. Function Execution Context:
- Whenever a function is invoked (called), a new function execution context is created for that function.
- The function execution context includes information about the specific function being called, such as its parameters and local variables.
- It also creates a new scope for the function, enabling access to variables defined within the function or its parent scopes.
- The function execution context is placed on top of the execution context stack.

3. Lexical Environment:
- An execution context has an associated lexical environment, which contains identifiers (variables and functions) available in the current scope.
- The lexical environment keeps track of the variables and functions defined within the current scope, along with their values or references.
- It also holds a reference to the outer environment (parent scope), allowing access to variables from outer scopes.

4. Scope Chain:
- The scope chain is a mechanism that determines how variable lookups are performed in nested scopes.
- Each execution context has access to variables in its own lexical environment and the outer environments.
- When a variable is accessed, JavaScript searches for it in the current execution context's lexical environment. If not found, it looks in the next outer environment until the variable is found or the global scope is reached.

an execution context represents the environment in which JavaScript code is executed. It manages variables, functions, and scope, allowing for proper execution *and access to data. The global execution context is created at the start of the program, while function execution contexts are created when functions are called. Each execution context has a lexical environment that holds variables and functions, and a scope chain that determines variable lookup.*



```
var n = 2;

function square(num) {
 var ans = num * num;
 return ans;
}

var square2 = square(n);
var square4 = square(4);
```