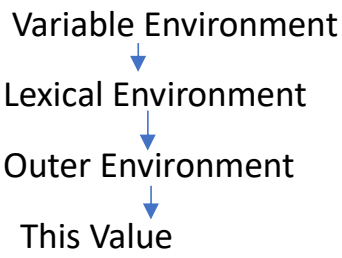
EXECUTION CONTEXT

In simple words, an execution context is like a container that holds all the information needed to execute a piece of JavaScript code. It manages variables, functions, and scopes during the execution of the code.

Here's a simplified diagram illustrating the components of an execution context:



Variable Environment: This component contains all the variables, function declarations, and function arguments defined within the context. It holds the values of variables and functions and allows access to them during code execution.

Lexical Environment: The lexical environment is similar to the variable environment but also includes an environment record. It keeps track of all the variables and functions defined within the context. The environment record is used for looking up variables and functions during execution.

Outer Environment: The outer environment refers to the lexical environment of the parent scope. It is used for variable and function resolution when the current execution context needs to access variables or functions from an outer scope.

This Value: The This value represents the context within which the current code is being executed. It refers to the object to which a method belongs or the global object (window in the browser) if the code is not inside any object.

During the execution of JavaScript code, the execution contexts are created and managed in a stack-like structure called the "call stack". When a function is called, a new function execution context is created and pushed onto the call stack. Once the function completes its execution, its context is popped off the stack, and the control goes back to the previous execution context.

Understanding the concept of execution context helps in understanding how variables, functions, and scopes are managed in JavaScript, and how code is executed step by step.