The Scope Chain, Scope and Lexical Environment

Wednesday, 16 August 2023 11:28 PM

→ Scope in JS is directly dependant on the lexical environment.

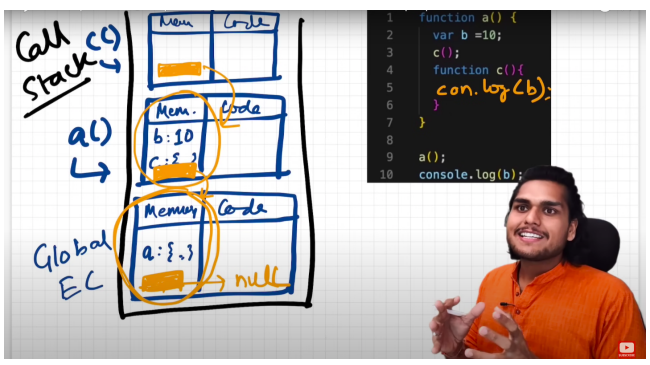
Scope: Scope basically means where you can access a specific variable or fⁿ inside the code.

Note: Wherever an execution context is created a lexical environment is also created.

Lexical Environment: It is the local memory + lexical environment of its parent.

→ Lexical means in hierarchy or in sequence.

Eg:



→ In this we can see that whenever an execution context is created, a lexical environment is also created.
→ fⁿ c() is lexically located inside fⁿ a(). fⁿ a() is the lexical parent of fⁿ c().

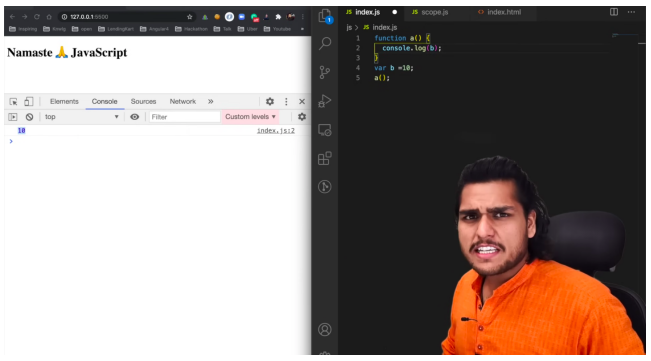
→ After all the memory creation phase of all ECs, when we reach line 5 in code execution phase, JS engine searches for the value of 'b' inside its local memory or local EC, since there is no 'b' inside the local memory of c(), therefore JS engine goes to the reference of the lexical environment of its parent i.e fⁿ c(), there it finds the value of b & prints it as 2.

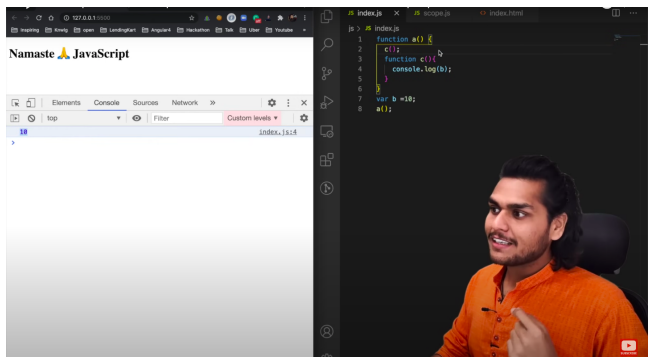
Note: If 'b' had not been declared anywhere then JS engine would have tried to find 'b' in the local memory of fⁿ a() and then it would have gone to the reference of the lexical environment of GEC and it would not have found

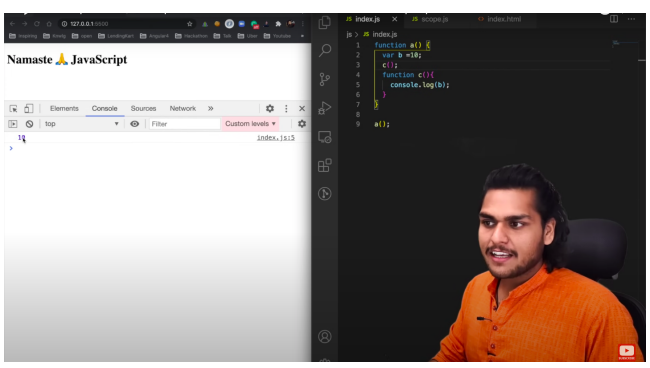
`b' there. So finally JS engine would have given reference error: b is not defined. Thus, we can say that 'b' is not in scope.

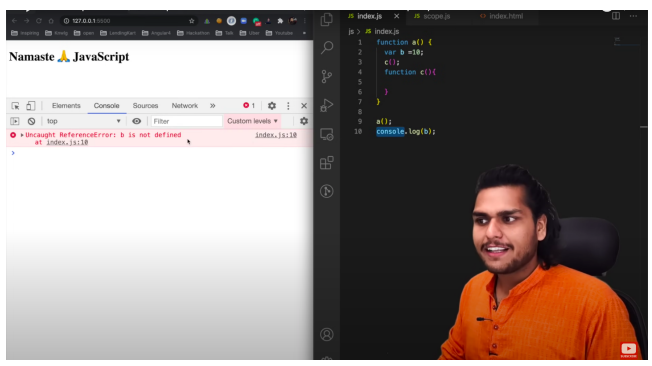
Note: The reference of the lexical environment inside the GEC points to null.

→ This is how we can justify the outputs of the following pieces of code:



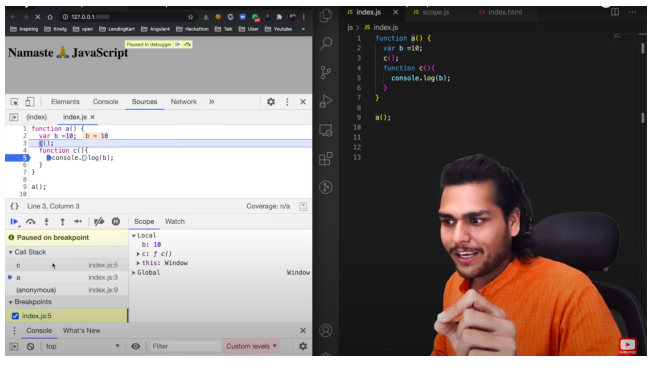




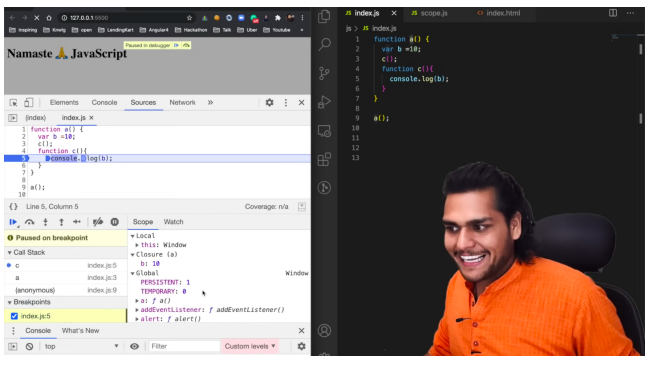


Scope Chain: It is the chain of the lexical environments and the parent references.

→ Browser demo showing the scope chain and lexical environments:



(local memory and lexical environment of fn a())



(local memory & lexical environment of fn c())