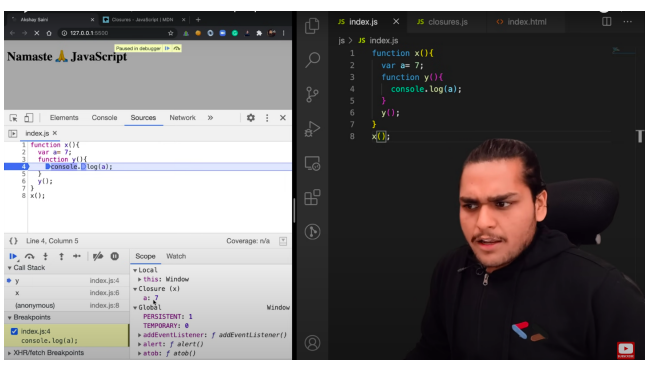
Closures in JS

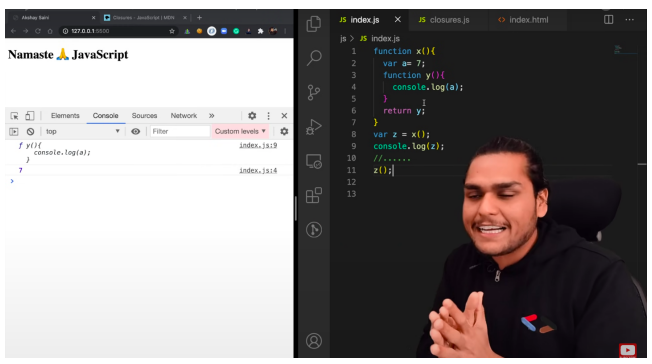
Thursday, 17 August 2023 10:49 PM

→ Closure: Closure is a fn bound together with its lexical environment. Therefore fn with its lexical scope bundled together is a closure.

eg:



→ In above the fn y() was bind to the variables of fn x().
Thus, it forms a closure & it has access to its parent's lexical scope.

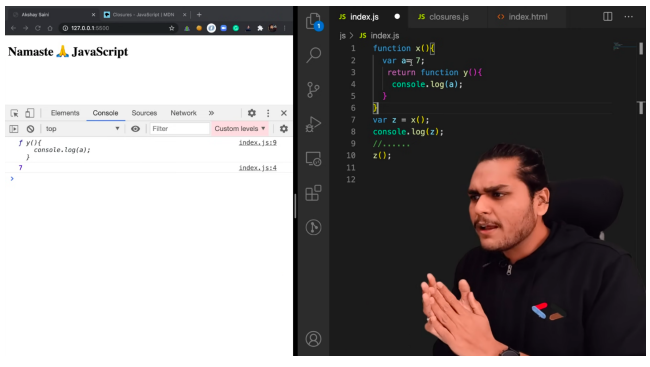


→ In JS we can return a fn, pass a fn as a parameter and whathot.

→ Explanation of above code & o/p:

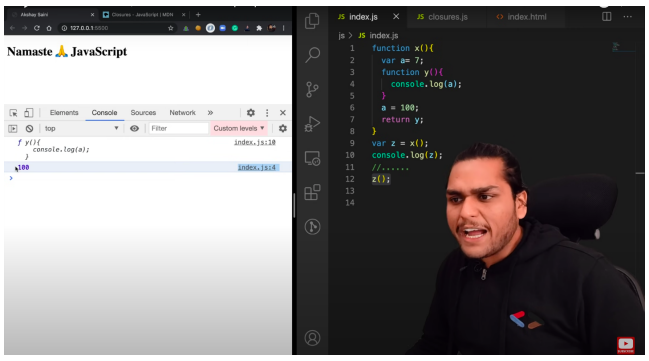
Now, we see that JS allows us to even return functions.
When a fn is returned, and stored inside a variable the whole fn code along with the reference of the lexical environment of its parent gets stored in the variable (in this case, z).
So, when we console 'z', we get the whole fn as the o/p.
Now, if we try to invoke z() anywhere in the program then we get o/p as 7.

Why is that?
This is because when y() was returned and stored in 'z', its reference to its parent's lexical environment was also stored in 'z'. Therefore a closure was returned & stored in 'z'.

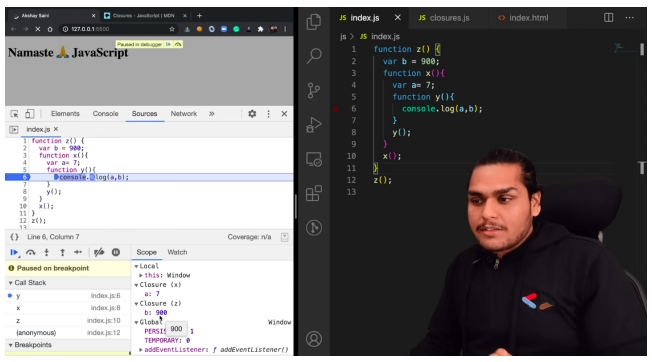


→ There is no difference b/w the above code and this code.

→ Corner cases :



→ Here, since the reference of its parent's Lexical environment is returned, thus, 100 is printed as the value of 'a' was changed to 100.



→ Here, fn y() forms closure with fn z() as well as fn z(). Thus, it has reference of lexical environments of both.

→ Uses of closures :

1) Module design pattern.
2) Currying
3) Functions like once
4) Memoize
5) Maintaining state in async world
6) SetTimeouts
7) Iterators and many more ...