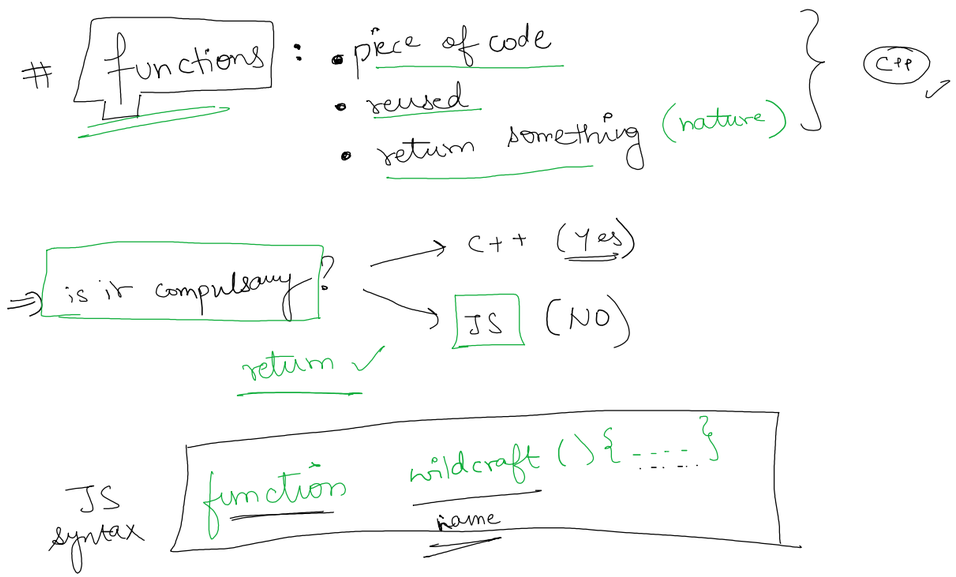
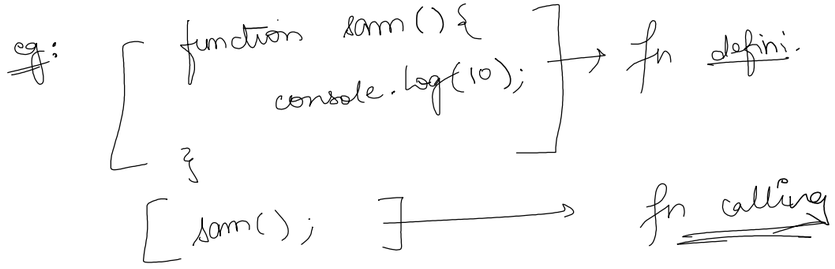
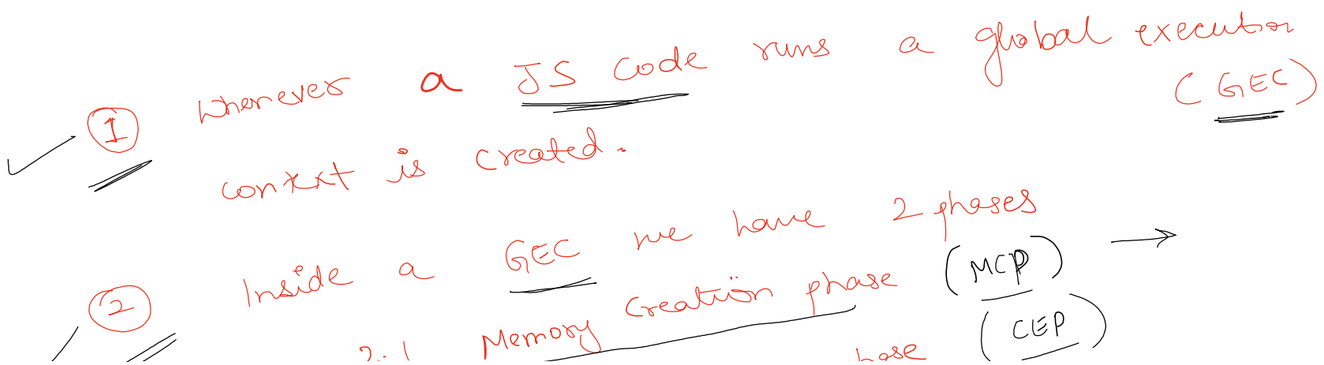
Day-15()

Saturday, 25 January 2025 5:08 PM







2.2 Code Execution pr~

(3) Role of MCP is to allocate memory
- 3.1 to all the variables
- 3.2 to all the fn's

{2}

(When 0 line of code has been executed)

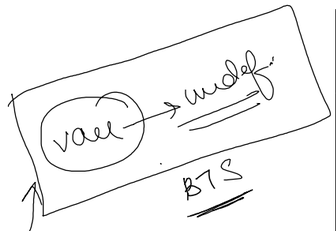
MCP runs before CEP *

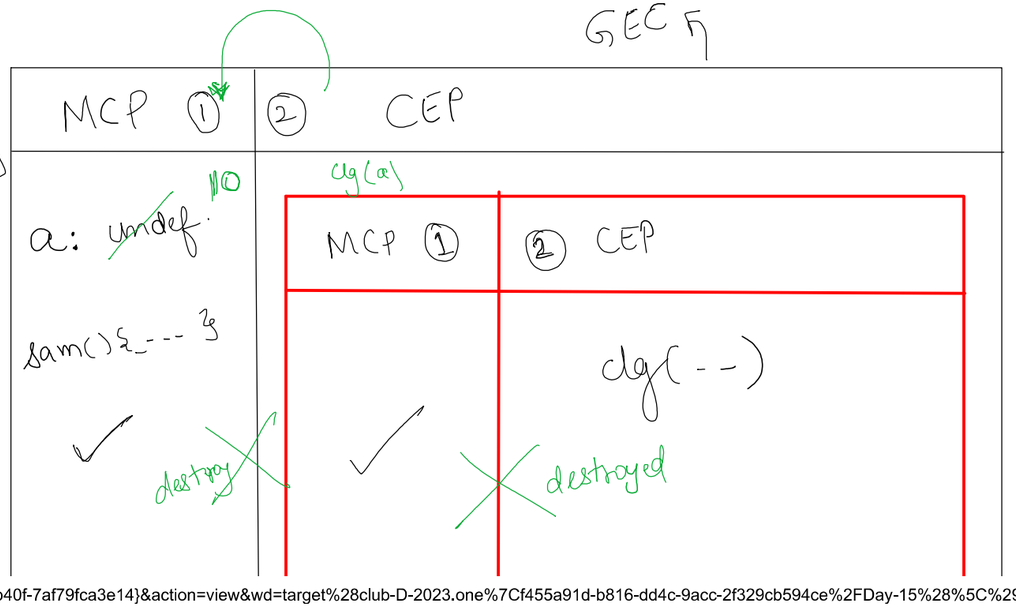
(4) CEP means running the code line by line after MCP is done. (0 line) → before that

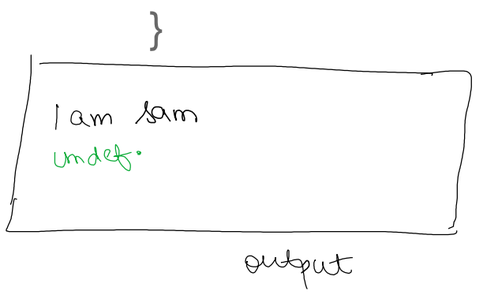
(5) Whenever a fn is called a new execution (EC) → MCP, CEP
context is created & the same process repeats.

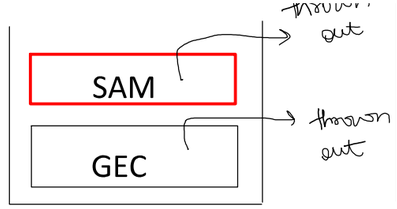
Later on (6) Whenever GEC is created, a global object along with it is also created. (window) ★

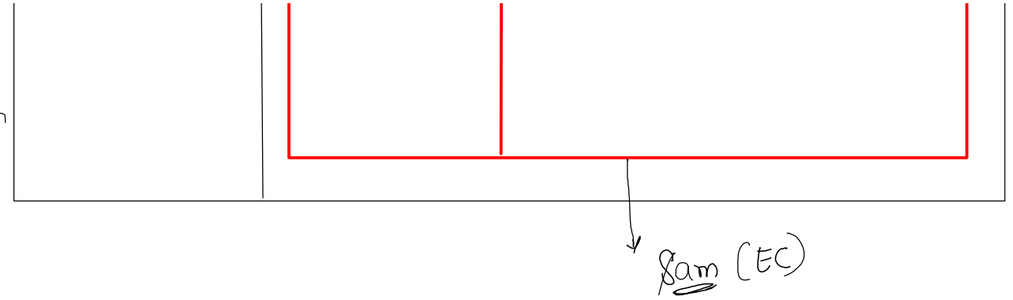
```
1  sam()
2  console.log(a);
3  var a = 10;
4  function sam(){
5      console.log("i am sam");
6
```



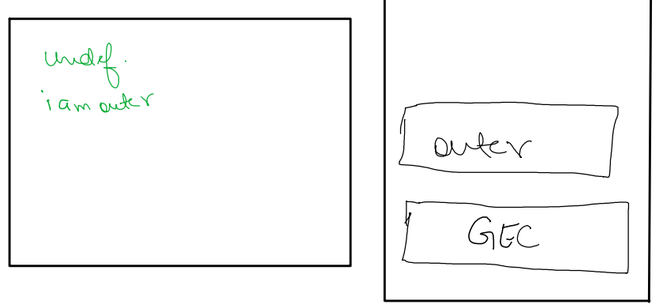


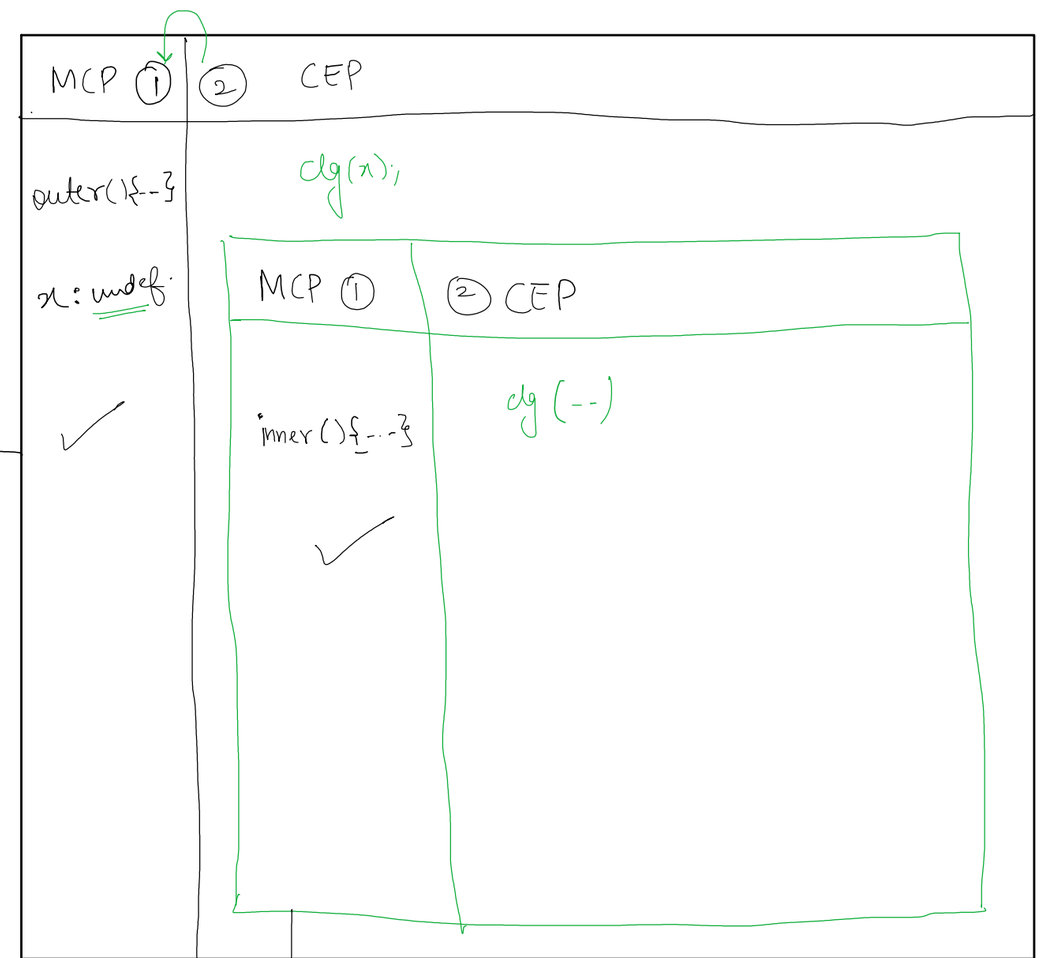


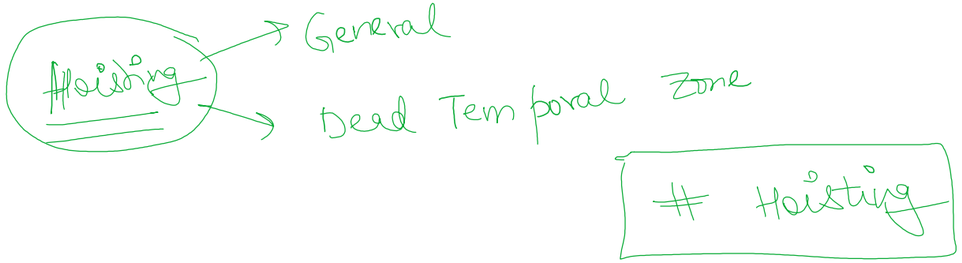





```
1  function outer(){
2      console.log("i am outer")
3      function inner(){
4          console.log("i am inner")
5          function veryInner(){
6              console.log("i am very inner");
7          }
8      }
9  }
10 console.log(x);
11 outer();
12 var x = 10;
13 console.log(x);
```

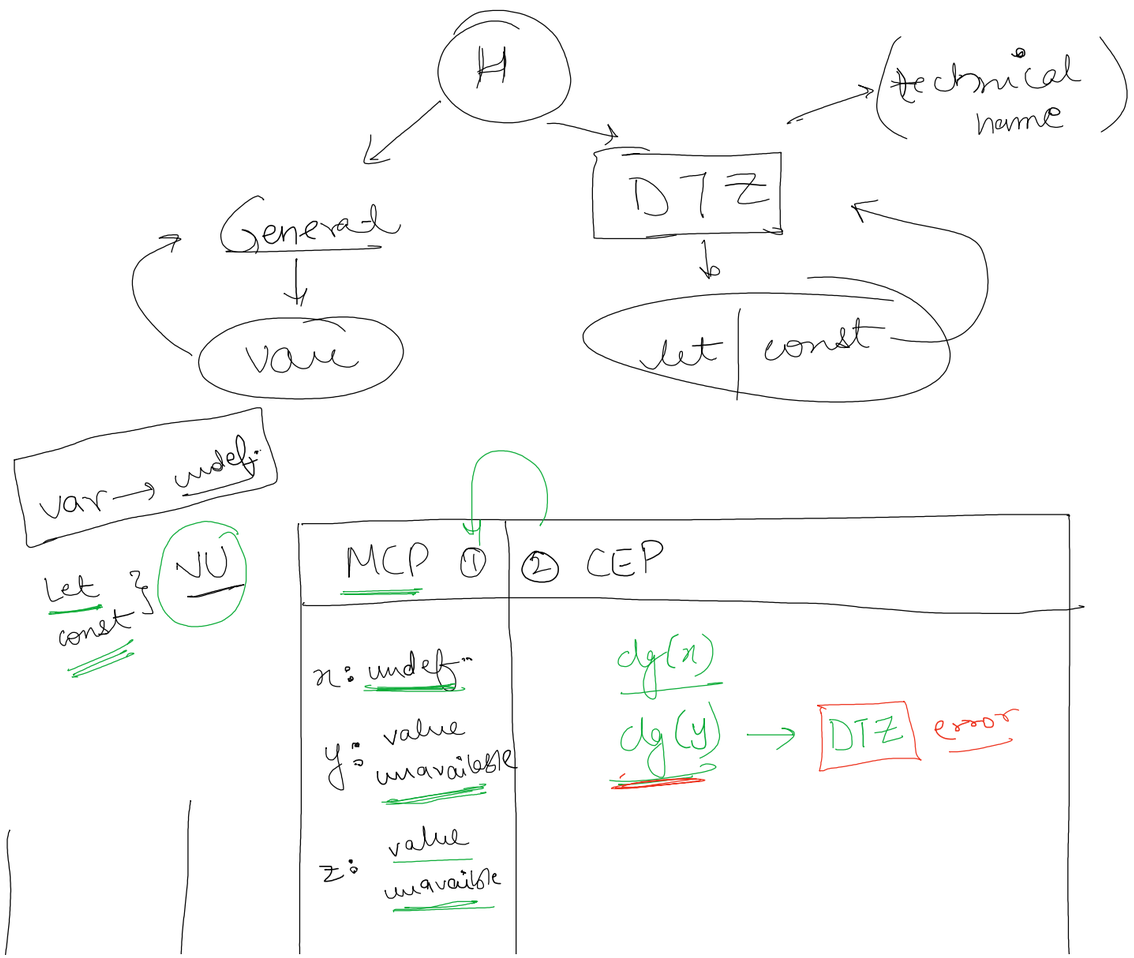




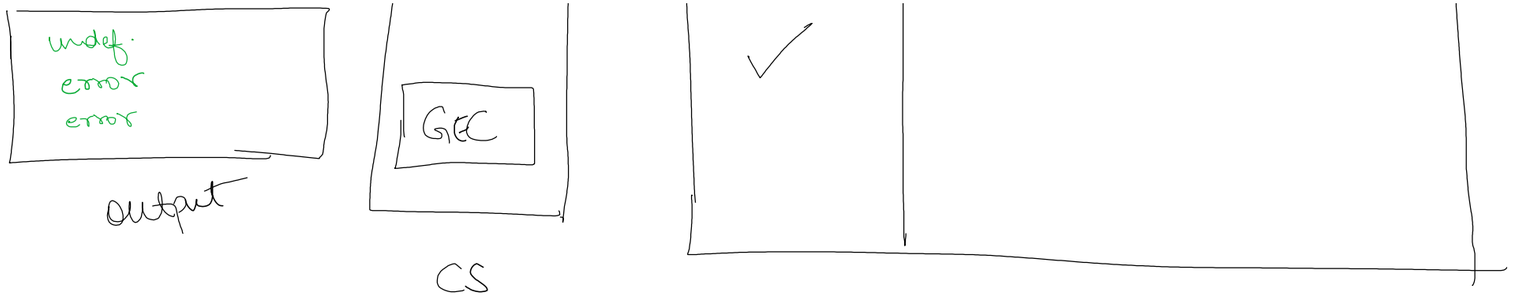


Hoisting

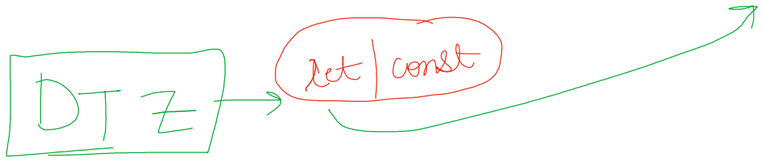
→ Concept of accessing a variable / fn even before its declared.

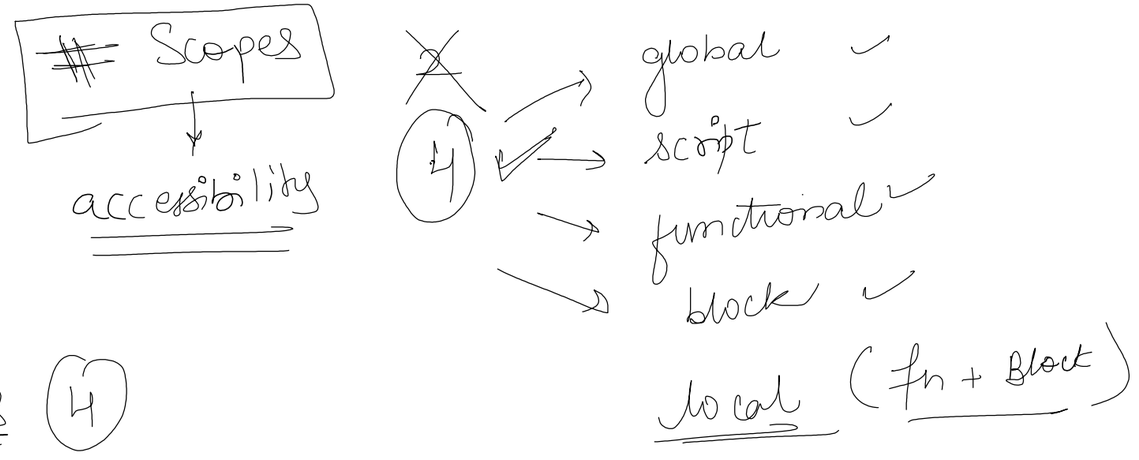



```
1  console.log(x);
2  console.log(y);
3  console.log(z);
4  var x = 10;
5  let y = 20;
6  const z = 30;
```



error → special error → you cannot access y|z before initialization





① global

② script

③ functional + ④ block ⇒ local

{ } → Block (technical name)

scope

Case 1 : is your variable inside " { } " ?

NO → var → global scope
let | const → script scope

YES → is that " { } " becoz of function ?

YES → var → functional scope
let | const → block scope

NO → var → global scope
let | const → block scope