JAVA Script

1) Declr of Variable :-

A) VAR → var a = 10

B) let

C) const

var :- 1) var has global scope & functional scope

2) we can redeclr variable with same name in the same scope & different scope

3) we can reinitialize the variable with same scope & different scope.

Exection Content :- (How to run JS internally)

JS

→ global execution content

↓

this = window

1) Memory allocation phase ✓

2) Execution phase ✓

JS

✓ → 1) global execution content

✓ → 2) function execution content

JS

↓

run

↓

global execution content

global execution content

window

1) Memory allocation phase

a : undefined

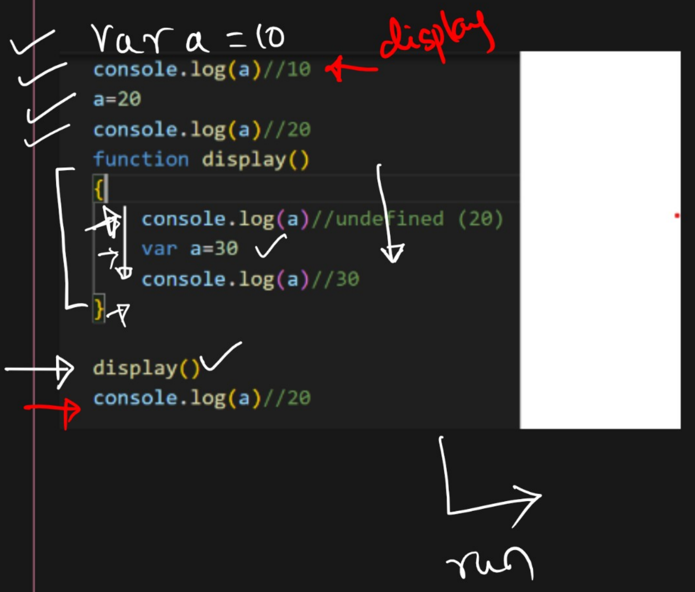
2) Execution phase

console.log(a) → undefined

✓ a : 10

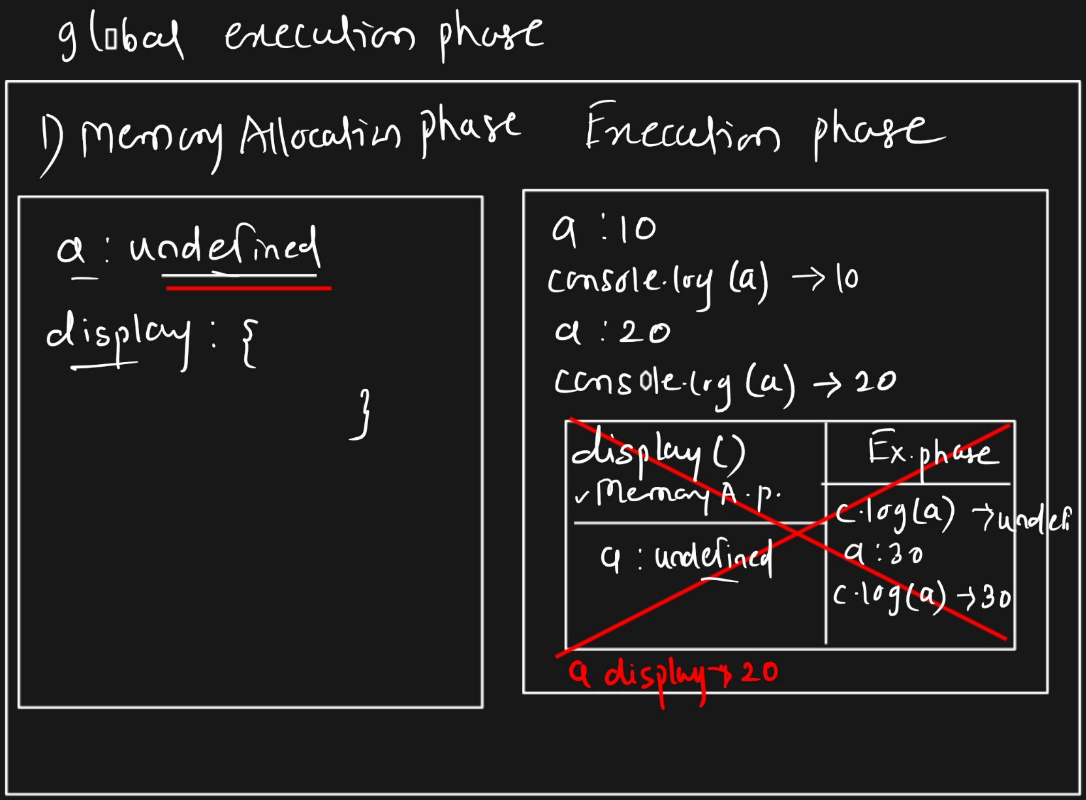
✓ Console.log($\overset{10}{a}$) → 10

```
→ console.log (a)
→ var a = 10
→ console. log (a)
```



```
var a =10
console.log(a)//10
a=20
console.log(a)//20
function display()
{
    console.log(a)//undefined (20)
    var a=30
    console.log(a)//30
}

display()
console.log(a)//20
```



Variable Hoisting :- It is behaviour of variable & funⁿ declⁿ are moving to top of their scope (either the global scope or the funⁿ scope) during the compilation phase before the code is executed.

```
console.log (a)
var a = 10
// scope global
```

Hoisting for var

console.log(a) $\Rightarrow$ undefined

var a = 10

let & const

console.log(a) → a is not defined

let a = 10

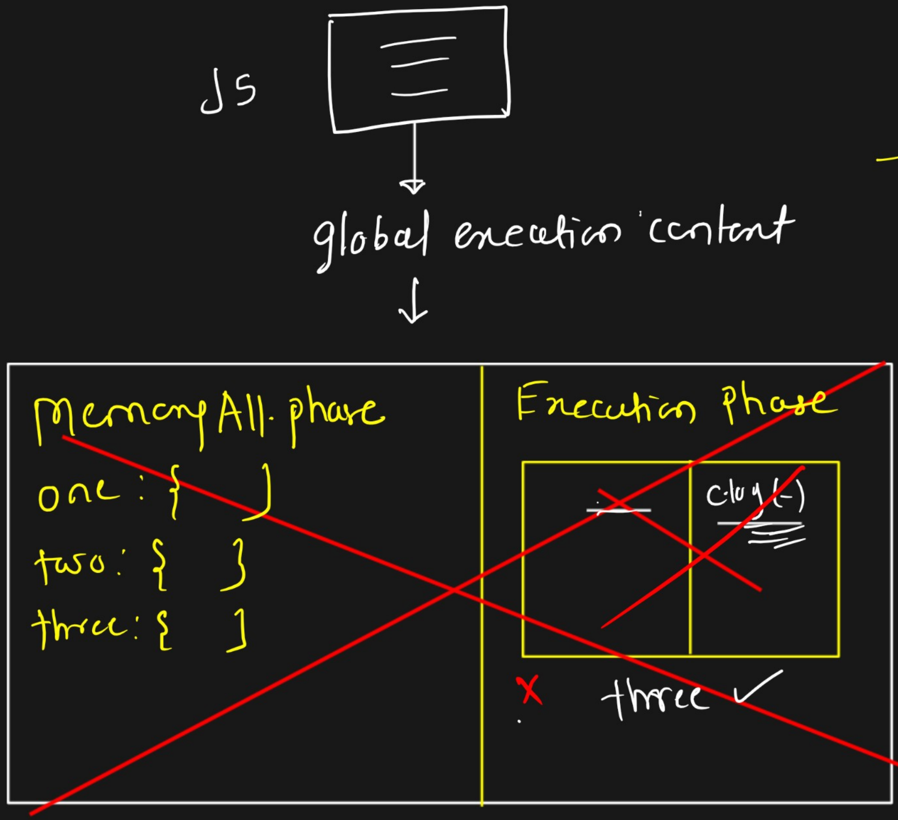
Whenever variable declr with let & const and try to access their value before their declr then it show's one error that is variable is not defined because it goes $\boxed{\text{temporal dead zone}}$

* Scope:-

1) global scope
2) funn scope
3) block scope
4) local scope
5) script scope

* Call Stack :-

JS

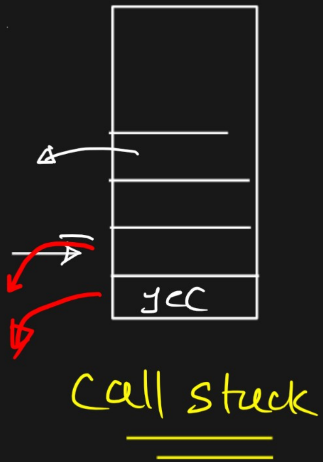


```
function one ()
{
→ c.log ("one");
}

function two ()
{
    c.log ('two')
}

function three ()
{
} c.log ('three'); ✓

→ one ()
→ two ()
  three ()
```



Call Stack :- They manage the all the operation of JS code

Variable Declr

2) let :- they have block scope & (script scope)
- redeclr - only in different scope
reinitialize - in same scope & different scope

3) const :- same as let type variable
as well as
reinitialize is not allowed

1) Variable declr → var, let & const diff
2) Scope variable → with code & browser
✓3) type of scope → global, local, block, script
4) execution content → global ~~expe~~ execution content ← M.A.P. exe.ph
5) Function execution content
6) Hoisting / Temporal Dead zone
7) call stack