Global Execution Context :

* Execution of the code.
* Store in two Phases
    - i) memory Phase
    - ii) Code Execution

Memory Phase: Store variable function name.

Code Execution: Assign values to variables & when function comes again Global Execution context memory & code Execution take Place :

Ex:

```
Var a ;
a = 10;
Var b = 20;
function hi ()
{
  Var a = 11;
  C.L (a);
}
C.L (a); // 10
```

Global EC

Memory	Code Execution

a : undefined
b : undefined } Before execution
 	
a = 10
b = 20 } After Execution
 	
hi : { hi()
{
var a = 11;
} C.L (a);
	a = 10
	
	b = 20 → function is a
	
	function comes (Local Scope)
	
	separate function EC
	
	Within the function this Memory Code Execution appears

function EC:

memory	Code Exec
a : undefi a = 11	a = 11;

Ex:-2

```
var a = 10;
var b = 20;
function Hello()
{
   return a+b;
}
C.L(a); // 10
C.L(Hello()); // 30
```

access because Global
if in function we can use within function because local Scope.

(because a, b are global scope)

memory	CE
a: undefin b: undefin a=10 b=20 hello	a=10 b=20 Hello()

Hello():

memor	C.E
a: un 10 b= un 20	a=10 b=20

Ex:-3

```
var d = 10;
var e = 20;
function a()
{
   var aa = "Hello";
   return aa;
}
C.L(a());
C.L(d);
```

Hello store in a()

Memory	C.E
d: un e: un a → function definition	d = 10 e = 20

memor	CE
aa: und Hello	aa=Hello

After function

Ex=3 Interpetur Steps

```
① { var a=10;
    var b=11;

    function ahello()
    {
      var a=122;
③     return a+10;
    }

② var h=ahello();

  C.L(h);  // 132
  C.L(a);  // 10
```

G.E.C

memory

a: undefined

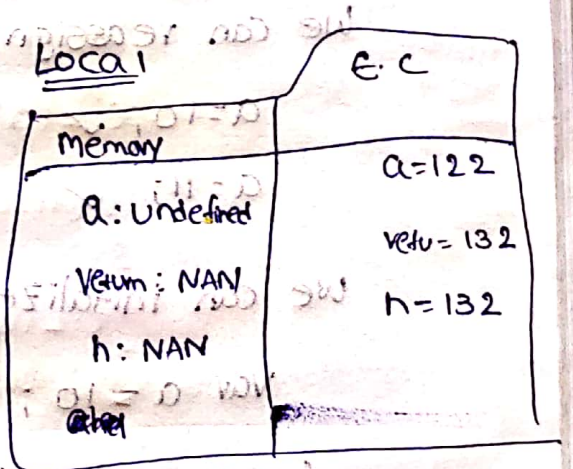
b: undefined

ahello: f ahello()

E.C

a=10

b=11



Var : It is a keyword which is used to declare a variable

Var, Let, const have two Phase; i) declaration Phase
ii) Scope Phase.

Var Let const
declare Scope.

Declaration Phase in var, let, const:

Var: we can declare.

var a; ✓

we can redeclare

var a; ✓

we can reassign.

a=10; ✓

a=11; ✓

we can initalize.

var a = 10;

we can re-initalize.

var a = 11; ✓

Var can done
Declartion Assignment initi

and we can do.

Redeclaration, Re-Assignment, Re-init

Disadvantage: var a = 10;

if reass a = 11;

[a will change.]

Before ES-6

To over come - let, Const

Let: we can declare we can redeclare.

Let a; ✓ Let a; ✗

we can reassign

a = 10;

a = 11; ✓

We cannot re-initalize

Let a = 10; → initalize ✓

Let a = 11; ✗ Re-intalize ✗

Const:

We cannot declare

Const a; ✗

We cannot Assign

Const a = 10; ✗

we cannot re assign

a = 11; ✗

We can do initialization.

Const a = 10; ✓

Const a = 11; ✗

	re-Declaration	re-assign	re-initalitation
Var	✓	✓	✓
Let	✗	✓	✗
Const	✗	✗	✗

Scopes in var, Let, Const :

Scopes will define : Where we have done the declaration.

Scopes : Global
Block
Local
Script

Var: default var is global scope.

Let, Const: Block scope.

Var: if we declare in global, outside the block we can use anywhere in code.

Global

```
var a = 10;
C.L(a); // 10
if (true)
{
  C.L(a); // 10
}
function hello()
{
  C.L(a); // 10
}
```

∴ because global scope.

if we use var in block we can use any where.

Block

```
if
{ var a = 10;
  C.L(10); //10
}
C.L(a); //10
```

var in global & block scope

Local : it is a functional scope.

Ex:
```
function name()
{
  var a = 10;
  C.L(a);
}
C.L(a); // not undefined.
```

Separate cont for function

While function running the variable is displayed & works After it doesn't work.

Var

Global	block	Local
✓	✓	X
We can use declare variable	We can use Outside when You mention in block bcoz var is global	We can't use because Local Create seperate Context

Let: block scope default

if we mention Let in global

Global act/ Script Scope

Ex: Let a = 10;

```
if (true)
{
  C.L(a);
}
```

* that Let will take script scope
* initailly Let is a block scope but if we declare it in global it will act as script scope and it will access in any where of code.

Block:-

```
if (true)
{ let a=10;
}
c.L(a);
```

∴ a is not defined

→ we can't access because let is a block scope we can access only in block.

Local:- function name()

```
{
  let a=10;
  c.L(a);
}
c.L(a);
```

Let is a block scope. can't access because, can't access outside. & function create separate content

Const:- defaut block.

Global Const a=10;

if we declare global in const script will act. access anywhere in code.

```
c.L(a); //10.
if (true)
{
  c.L(a); //10
}
```

block:-

```
if (true)
{ const a=10;
}
c.l(a); // can't access because block scope
        not defined.
```

local:-

```
function name()
{
  const a=10;
  c.l(a);
}
c.l(a); // not access because it create
        seperate context
```

Global scope = block = Local

Let

it take script

✓ we can use variable when we declare global it take script context

✗ in block can't access outside

✗ in function block create seperate context

Const

✓ we can use variable when we declare global

✗ in block

✗