→ Day-4 Functions

→ functions are the ❤ of JS.
→ main building block of the prog.

→ when ever a new function is called a new execution context is created for that function inside the stack/ call stack.

Q. What is execution context?
→ it is a container like structure which is divided into 2 phase.
(a) Memory execution (b) Code execution

☆ In this the execution of the code happen.

→ Every thing in JS happen in execution context.

☆ Call Stack

- It's a stack like structure, which shows the execution of program.
- At, it's bottom it has the global execution content.
- As soon the program finish it's execution the execution content is poped out from the stack.

← some important terms in functions

(A) Function decleration

```
func show(){
  //Something
}
```

(B) function call

```
fun show();
```

(C) Argument & parameter

```
fun Add(num1, num2);
// here num1, num2 are argument
fun Add(num1, num2){
return num1+nw2;
}
//here num1, num2 are parameter
```

(D) Func^n expression

```
var b= func(){
alert("Hi");
}
```

(E) Arrow function

```
let sum=()=>{
//Something
}
```

→ Diffrence b/w funcn expression & funcn decleration:

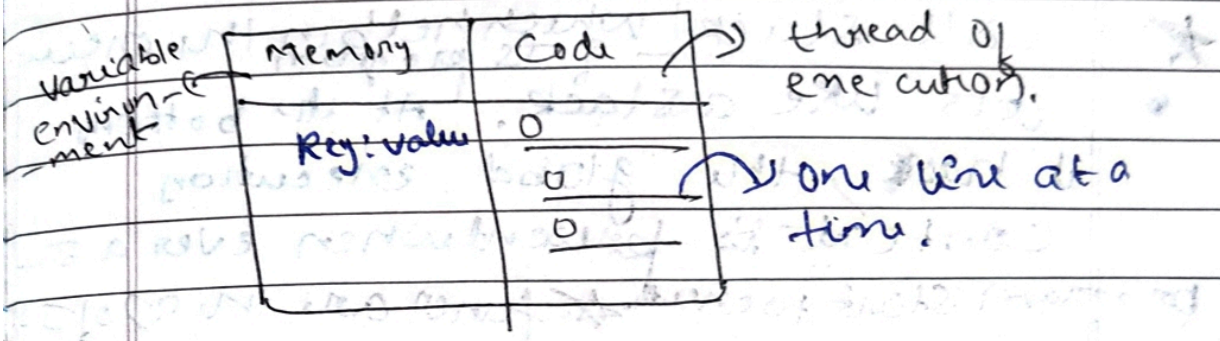
- A funcn decleration can be called earlier than it's defined but in case of expression this is not possible.

JAVASCRIPT

Page No.
Date

→ Every thing in JS will happen in execution context.

execution context



→ JS is synchronous & single threaded lang.

→ When a code is enter then a global execution content is formed in which the memory creation phase has all the variable which are in our prog. & assigned a special value undefined.

→ Now when all the variable are written in the memory allocation phase then code execution phase will run & after all the variable in the variable execution phase will get replaced by the value of the variable.

→ whenever any funcn came a new execution content is created for that

When the code comes return keyword it will give back a value to variable which envoked the funct

Page No.
Date

↳ When all the code is execution then the Global execution content is deleted.

Whenever a JS code is start executing line by line then a global execution content is created in which it has two phase memory phase & code executions phase. In the memory phase all variable are stored & assigned a special value of undefined

★ Call stack
- It is like a
it has, the
content is
program start to exe
- Whenever
any new
is created then it
onto that stack.
- As soon as the newly encu
function is executed the
newly created content is
deleted from the call stack.

☆ It is only to manage the E.C of JS

- **Hoisting in JAVA SCRIPT**

→ Accessing the variable before ~~utilize~~ utilizing them is known as hoisting.

☆ undefined & not defined

→ Even before the code start executing memory is given to each variable & function.

→ It allocate memory to each variable before a single line of code is executed

→ JS is a loosly typed language
i.e → you can put number into a variable after initializing it with a string

↳ Not defined is like you are trying to acess a variable which is not even given memory by the JS engine & undefined is the memory is allocated to that variable

~~Map, filter & Reduce~~

→ Scope Chain & Lexical Environment

☆ Scope mean where we can acess our variable & function.

the c will look at the a in his local memory if still not found then it will search in the parent of a.

↳ Lexical Env
memory alo
of its par
hierarchy]

↳ lexical mea
is present

```
function a(){
  var b=10;
  c();
function c(){
  console.log(b);
}}
a();
```

so, we can say that the function c is lexically inside a.

→ Lexical Scope is
access variable from

Let & Const in JS

. Let & const can be hoisted.

☆ let & const are stored in different memory space.

Scope in JS

→ It is a region of the execution where the value & expression can be referenced.

→ Scope determine the accessability of variable.

(a) Block scope (b) function scope

(c) global scope.

(c) → In this the variable can be accessed from any part of the code.

(a) → variable declared inside a {} can't be accessed outside the block.

(b) → variable declared inside a function, became local to the func

→ variable declared inside a function can't accessible outside

ZEN TECHNOLOGIES LTD

DATE / /

MON TUE WED THU FRI SAT SUN

→ TDZ (Temporal Dead Zone)

~~the undt~~ its the time when the let variable is housted & its initilized a value.

Page No. ____ Date ____

↳ When ever we are trying to acess any variable in Temporal deadzone it will give us refrence error.

→ When variable are in TDZ you can't acess the variable, only be acess if they are

↳ diffrence b/w Syntax, type & Refrence error

- Type error → can't intilize after declaring
- Syntax error → didn't intilize any thing to the const variable
- Refrence variable: ~~ea~~ you are acessing the variable before initilising it to a number or anything

→ **Scopes in variable**