Day 3

```
In js we have two categories of  types :
```

value type aka primitive and reference types

Value type : Number , String , Boolean , Symbol , undefine and null
Reference types : Object , function and arrays

Primitives are copied by their value and objects are copied by their references

for eg :

```
let x = 10;
let y = x;

x= 20;

//output shall be : x = 20 and y = 10 i.e. they are independent
```

10 is stored in the x variable and then the value of the x variable is copied to the y variable . hence they are completely independent

now , lets see another example where we are changing the x into an object :

```
let x = { value: 10};// x is an object with a property known as value
let y = x;

x.value= 20;

//output shall be : x = 20 and y = 20
```

When we make an object , that object is not stored in the variable 'x'
the object is stored somewhere else in the memory and the address of that memory location is stored inside that variable . so when we copy the x to the y , it is the address or the reference that we are copying . that is both the x and y are pointed to the same object in memory , and when we modify the object by using either x or y ,the changes are immediately visible to the other variable .

another example :

```
let obj = { value : 10 };

function increase (obj){
        obj.value++;
}

increase(obj);
console.log(obj);

//the value is increased by one
```

Scope (Local and Global)

Scope in JavaScript refers to the accessibility of variables and functions within your code. It determines where you can use a particular variable or function and how it interacts with other parts of your program. There are two main types of scope in JavaScript:

1. Global Scope:

- Variables declared outside of any function or block are considered to be in the global scope.
- They are accessible from anywhere in your program, which can lead to naming conflicts and make your code harder to maintain.

Example:

JavaScript

```
let globalVar = "This is a global variable";

function myFunction() {
  console.log(globalVar); // Can access global variable from within the function
}

myFunction();
console.log(globalVar); // Can access global variable outside the function
```

2. Local Scope:

- Variables declared within a function or block (like an if statement or a loop) are considered to be in the local scope.
- They are only accessible from within that specific function or block, preventing conflicts with variables of the same name in other parts of your code.

JavaScript has two main types of local scope:

- **Function Scope:** Variables declared with `var`, `let`, or `const` within a function are in function scope. They are only accessible from within that function.
- **Block Scope:** Variables declared with `let` or `const` within a block (like an if statement or a loop) are in block scope. They are only accessible from within that specific block.

Example (Function Scope):

JavaScript

```
function myFunction() {
  let localVar = "This is a local variable";
  console.log(localVar); // Can access localVar within the function
}

myFunction();
// console.log(localVar); // This would cause an error because localVar is not
defined here (outside the function)
```

Example (Block Scope):

JavaScript

```
if (true) {
  let blockVar = "This is a block variable";
  console.log(blockVar); // Can access blockVar within the if block
}

// console.log(blockVar); // This would cause an error because blockVar is not
defined outside the if block
```

Key Points:

- Function arguments also have their own local scope within the function.
- `var` has a different scoping behavior within loops compared to `let` and `const`.
- Understanding scope is essential for writing clean, maintainable, and bug-free JavaScript code.

Tips:

- Use `const` by default for variables that don't need to be reassigned.
- Avoid using global variables whenever possible.
- Use descriptive variable names to avoid conflicts.

Var vs Let vs constant

Feature	var	let
Scope	Function scope	Block scope

let, const => block scoped
var =>function scoped

This keyword :

```
const video ={
        title : 'a', // this is the property of the video object
        play(){
                console.log(this);
        }
}

video.play();
```

```
function playvide(){
        console.log(this);
}

playvideo();
```