DAY - 1

```java
const message = 'Hello world!'
var a=2
console.log(message)
console.log(a)
function myfunc(){
  const mes='Function printed'
  var b=2
  var c=3
  var d=b+c
  console.log(d)
  console.log(mes)
}
myfunc()
function calc(x=5){
  if (x%2==0){
    console.log('Even')
  }
  else{
    console.log('odd')
  }
}
calc()

console.log(`my message is ${message}`) // looks pretier than the next line
console.log('my message is ',message) //looks ugly compared with the previous line
```

contains basic things of JavaScript and 24th line is the instatement line where it is clean compared with the normal console.log

	Data types	Assignation	Declaration	Scope of the variables/data types inside the function
1	let	reassign	redeclare	function scope
2	var	reassign	no	block scope
3	const	no	no	block scope

```java
//reassign code
var a=10
a=20//reassign

let b=30
b=40//reassign
```

above is the reassigning of the values and variables

```java
var name='sai'
var name='ravi' //redeclare of the variables
function disp(){
    name = 'ram'
    console.log(`i am inside func ${name}`)
}
console.log(`i am outside func ${name}`)

//function called
```

```
disp()


/*
output is
i am outside func sai
i am inside func ram
*/
```

in const redeclare is not possible it throws errors

```
let name='dev'
let name='darshan' // throws error
name='darshan' //not throws error
function mu(){
    let name ='devdarshan' //not throws error
    console.log(`My name is ${name}`)
}
console.log(`My name is ${name}`)
mu()
```

'let' can be reassigned but redeclaration can't be done

Hoisting

Hoisting in JavaScript is a built-in behavior that moves the declarations of functions, variables, and classes to the top of their scope before the code is executed.

```
function sam(){
    b=10 //without data type wont throw error
}
sam()
b=20 // can over ride the value
console.log(b)
```

Array Operations

```
const a=[20,30,40]
console.log(a) //prints the values
a[1]=50 //changes the value of first index
console.log(a)
```

In the arrays the const values also can be changed

```
const dict={'name':'dev','age':19} //dictionary
dict.age=20 //change using keys
dict.name='Devdarshan'
console.log(dict)//prints the dictionary
```

```
const a=[20,30,40]
const b=[50,60,70]
const c=a+','+b //combines two arrays
console.log(c)
```

```java
const para=`devdarshan
is a good boy and proper boy` // ` prints n no of lines inside it
console.log(para)
```

For big number of input (big int) we should add 'n' to the end of the number so that we can print the big input number

```java
let a=[20,"Dev",40]
let dates=[100,200,300]
let person={name:'deva',roll:23}
a.push(dates,800,person)
console.log(a) //returns all the elements including a,dates,persons
```

push() used to add elements or to combine n number of lists, arrays, etc. Adds everything in the back of the array

```java
let a=[20,"Dev",40]
let dates=[100,200,300]
let person={name:'deva',roll:23}
a.unshift(dates,800,person)
console.log(a)

//output
//[ [ 100, 200, 300 ], 800, { name: 'deva', roll: 23 }, 20, 'Dev', 40 ]
```

unshift() means adds the data to the front of the array

```java
let a=[20,"Dev",40]
console.log(a.pop()) // pops the last element
//if we want to pop an specific element we use
console.log(a[1].pop() //removes the 1st index element
```

```java
let a=[20,"Dev",40]
console.log(a.shift()) // removes the first element of the array
a.length // retruns the length of the array
```

Delete Function

```java
let a=[20,"Dev",40]
console.log(delete a[1])
console.log(a) //[ 20, <1 empty item>, 40 ]

for (let i=0;i<a.length;i++){
    console.log(a[i])
}
/*output of for loop is 20
undefined
40
*/
```

splice(index_no, no of elements we don't want, replace any other element)

```java
let a=[10,20,30,40,50]
a.splice(1,2,'sai')
console.log(a) //[ 10, 'sai', 40, 50 ]
//removes 2 elements adds sai at the second index
```

slice(start index, End index) it gives the value form the starting index till the before value of the end index

```java
let a=[10,20,30,40,50]
console.log(a.slice(1,4)) //[20,30,40]
```

Warnings

```java
const a=2345678919882991879998929822 8n //syntax for big int values
console.log(a)
```

```java
console.log(5<Infinity) // returns true remember the syntax for infinity i always caps
```

Type of()

```java
let a=true
const b=false
console.log(typeof(a)) //boolean

a=0 //number
a=23.3222 //number
a='dev' //string a=[20,30,40] //object
a=null //object let c
console.log(c) //undefined
```

== means checks only the input values

=== means checks the value and the data type also

```java
var a=1
var b='1'
console.log(a==b) //true
console.log(a===b) //false
```

** is used to calculate power

```java
console.log(8&3) //0
console.log(8|8) //8
console.log(8^8) //0
```

Enhanced for loop

```java
let a=[10,20,30,40,50]
for (let data of a){
    console.log(data)
}
```