KONGU Day 2

Conditional statements / Decision making / Control statements:

```
a = 10
if(a%2 == 0){
  console.log("Even", a)
}
else {
  console.log("odd", a);
}
```

)

operators

1. Assignment operators (=)
2. Arithmetic operators (+, -, *, /, %)
3. Relational operators (<, >, <=, >=, ==, ===, !==, !==) ✓ ✓
   ✓ value with datatype
4. Logical operators (&&, ||, !)
5. Bitwise operators (&, |, ~, >>, <<, ^)
6. Ternary operator (? :) ⇒ ES6
7. Unary operator (++, --)

Secondary data types:

1. Array ✓ ⇒ arr = [a, 'b', 'c'] / [10, 20, 30] / [10, 'a', "abc", [20, 30]]
2. object ⇒ {a : 10, b : 20}

looping:

a = 0
while(a <= 10) {

```
for(a=0; a<=10; a++){
    clg(a);
}
```

```
while(a<=10){
    clg(a)
    ++a;
}
```

```
do{
    clg(a)
    a++;
}while(a<=10);
```

① for of

```
for(var a of arr){
    clg(a)
}
```

arr → [10, 20, 30]

```
o/p: 10
     20
     30
```

② for in

```
for(var a in arr){
    clg(a)
}
```

arr → [10, 20, 30]

```
o/p: 0
     1
     2
```

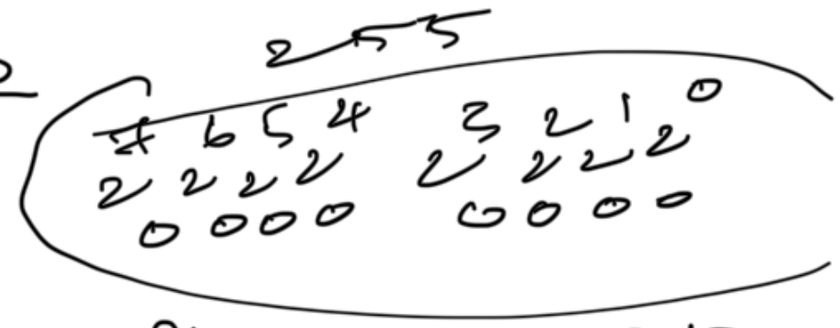
&

A	B	A ∩ B
0	0	0
0	1	0
1	0	0
1	1	1

|

A	B	A ∪ B
0	0	0
0	1	1
1	0	1
1	1	1

① Logical operator (&)




```
rel1 = 10
rel2 = 15
console.log(rel1 & rel2);
```

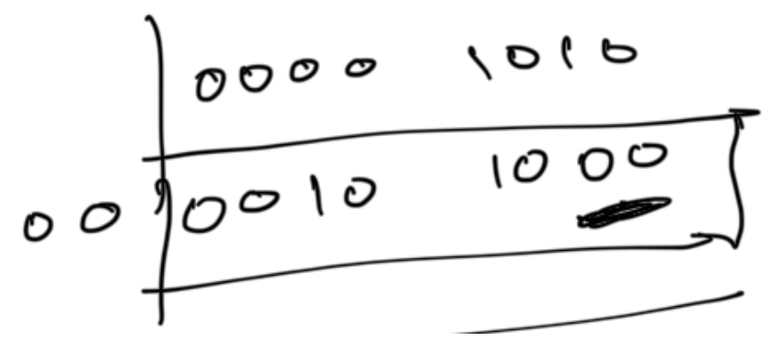
rel1 = 0000 1010
& rel2 = 0000 1111
0000 1010

→ 10

② bitwise operator (>>)

```
bit1 = 10
clg(bit1 >>2) o/p: 2
clg(bit1 <<2) o/p: 40
```

write a program to find even or odd without using modules operator?
use bitwise operator



Hint:

Task 1 ∈ Day2

32 + 8 ∈ 40

functions Syntax:

```
function funEg( ){

}
```

Types of functions

1. function without argument with return type / w/o rt
2. function with argument with rt / w.o rt

```
function KEC (students ){
```

```
clg(students)

}
KEC(["CSE", "IT", "AIML"])
```

o/p: ['CSE', 'IT', 'AIML']

ES6

1. variables → var, let, const
2. arrow Function.

```
arrowFun=()=>{
    console.log("Hello, KEC")

}
```

arrow fun( )

③ Destructing operator

var marks = [90, 95, 93]

var [m1, m2, m3] = marks

clg (m1)

clg (m2)

clg (m3)

④ Ternary operator.

a = 10

out = (a%2 == 0) ? "Even" : "odd"

clg (out).

⑤ Spread operator.

studentsList = ["Tom", "Cruise", "Holland",
"Tony", "Stark", "Robert",

"Downey", "Sr"]

newStudent = ["Dinesh", "Anand", "Ajay", "Madhav"]

CombinedStudentsList: [...studentsList, ...newStudent]

clg(combinedStudentsList)

o/p:

(6)

Task3: rest operator (...variables)

function restopeg(a, b, ...c) {

}

(7) Scope

1) Local Scope | Block Scope

② Global Scope

```
const pi = 3.14
var a = 5
clg(pi) ✓
clg(a) ✓

{
  let a = 10
  let b = 10
  clg(a) ✓
  clg(b) ✓
}
clg(a) ✓
```

op:

3.14
5
10
10
5

(8) Hoisting.

(1) clg(a)
var a = 10
clg(a)

o/p:
undefined
10

(2) clg(a)
const a = 3.14
clg(a)

~~undefined~~ Reference Error
3.14

(3) clg(a)
let a = 10
clg(a)

~~undefined~~ Reference Error
10

× Intermediate ×

callback :)

↳ Nested function

```
function submitForm(formMsg, dbs) {

    dbs()

    clg(formMsg);

}

function databaseStoring() {

    clg("Data stored successfully")

}

submitForm("Form Submitted Successfully", 'database Storing')
```