Agenda

1. Lexical scope in js
2. Closures, nested closures
3. Closures vs block scope
4. Application of Closures
5. HOF.

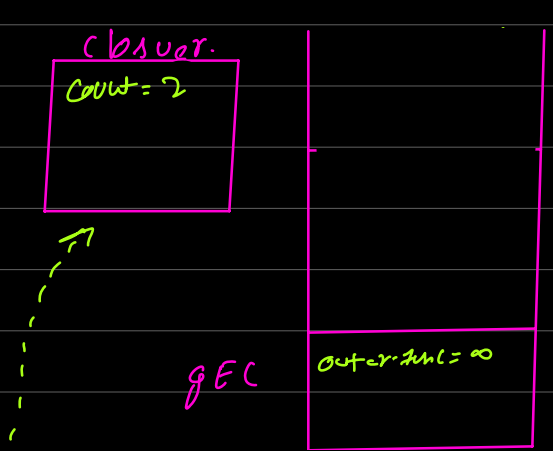
file1

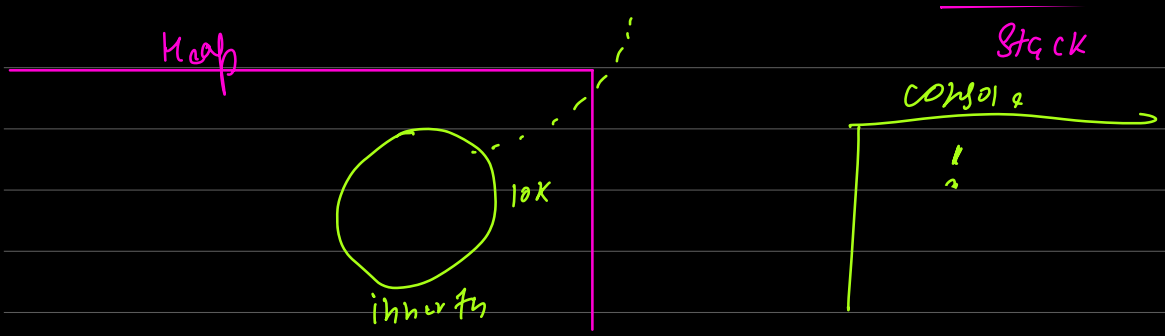
```
var x = 10
function () {
log (x)
}
```

file2

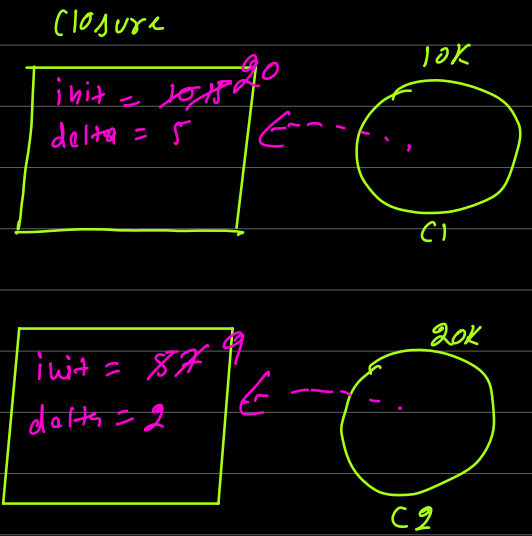
```
var x = 20
fn();
```

```
function outerFunction() {
    let count = 0;
    function innerFunction() {
        count++;
        return count
    }
    return innerFunction
}
const innerFunc = outerFunction();
console.log(innerFunc())
console.log(innerFunc())
```

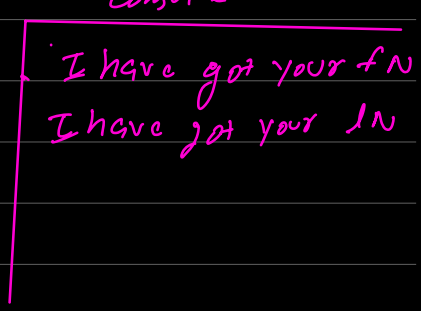


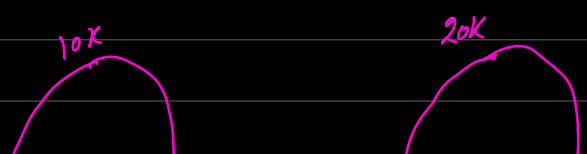


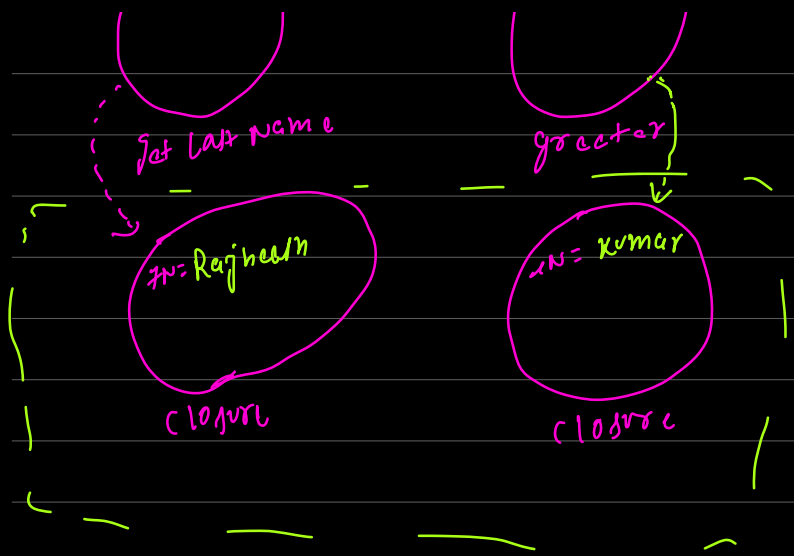

```
function createCounter(init, delta) {
    function count() {
        init = init + delta;
        return init;
    }
    return count;
}
let c1 = createCounter(10, 5);
let c2 = createCounter(5, 2);

console.log(c1()) 15
console.log(c2()) 7

console.log(c1()) 20
console.log(c2()) 9
```

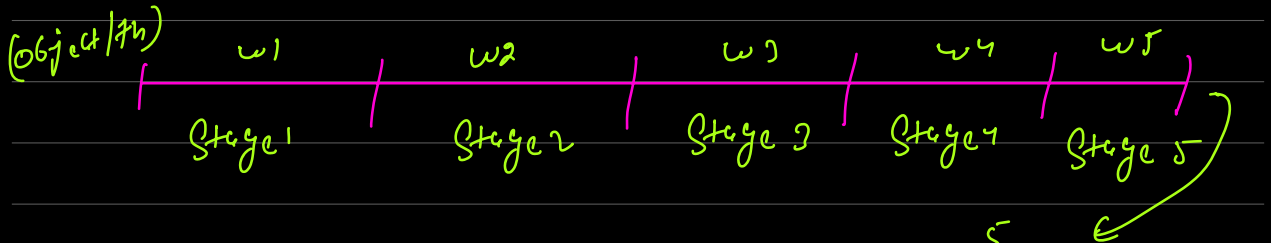



```
let iamINGEC = 200;
function getFirstName(firstName) {
    console.log("I have got your first Name");
    return function getLastName(lastName) {
        console.log("I have got Your last Name");
        return function greeter() {
            console.log(`Hi I am ${firstName} ${lastName}`);  // closure
            console.log("Hi GEC", iamINGEC) // LExical scope
            iamINGEC++;
        }
    }
}

const fnNameRtrn = getFirstName("Rajneesh");
const lnNameRtrn = fnNameRtrn("Kumar");
lnNameRtrn();
```



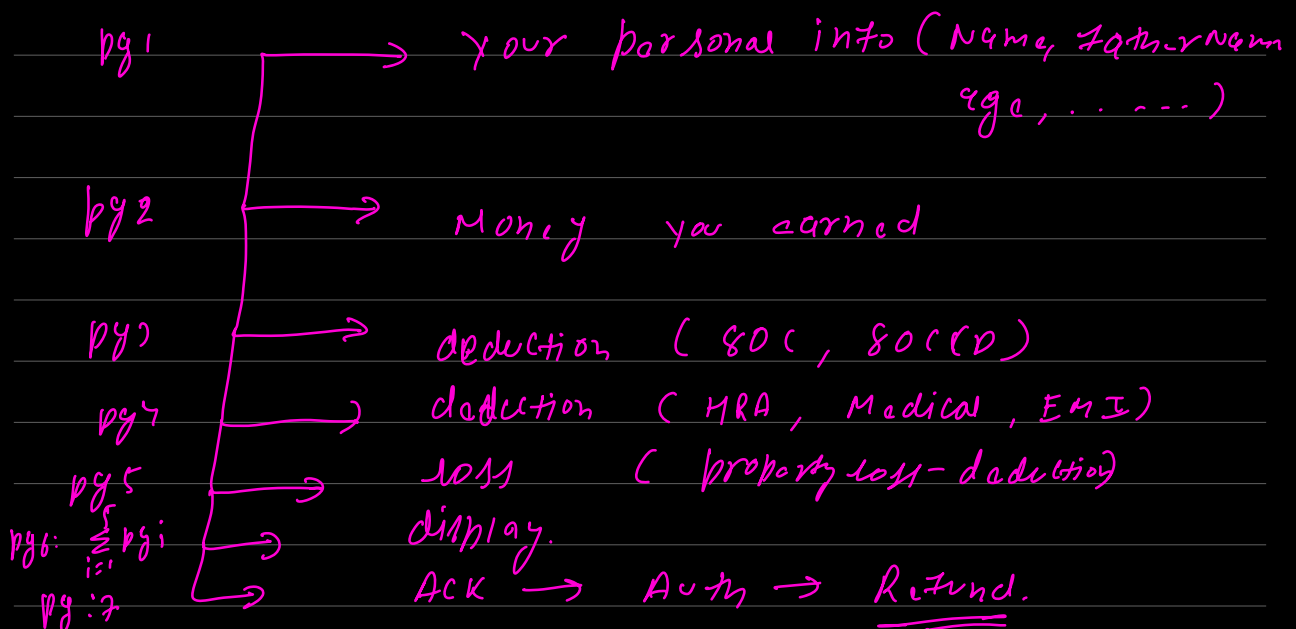




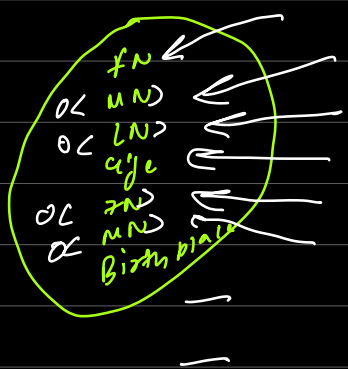


$\sum_{i=1}^{5} w_i$

(Income Tax Portal of India)



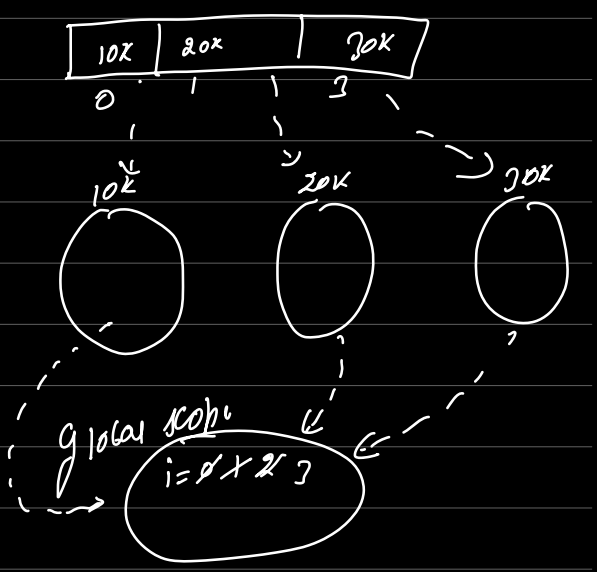
(personal Info Builder)



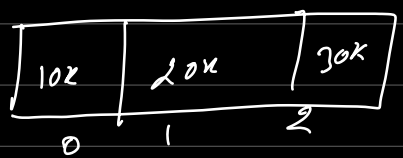
Rajneesh Kumar yadav
Rajneesh Kumar

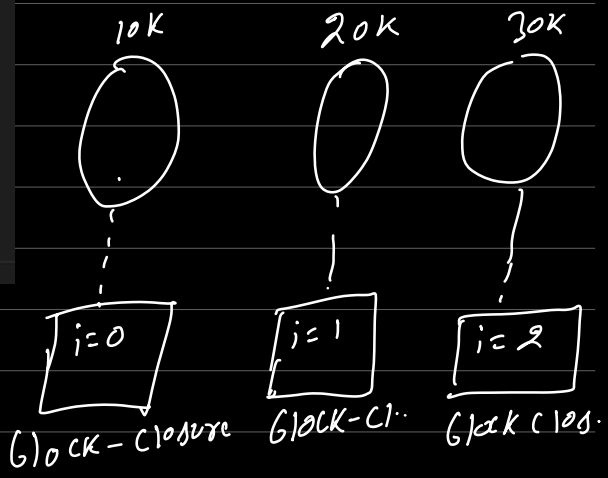
(fN). (10N). (_). ( )

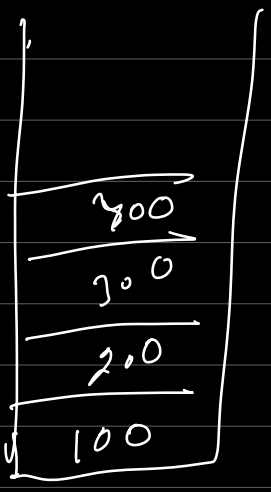
```
function outer() {
    let arrFn = [];
    for (var i = 0 ;i < 3; i++) {
        arrFn.push(function fn() {
            i++;
            console.log(i);
        })
    }
    return arrFn;
}

let arrFn = outer();
arrFn[0](); 4
arrFn[1](); 5
arrFn[2](); 6
```



```
function outer() {
    /**
     * arrfns block scope refer to -> functions
     * */
    let arrFn = [];
    for (let i = 0; i < 3; i++) {
        arrFn.push(function fn() {
            i++;
            console.log(i);
        })
    }
    return arrFn;
}
let arrFn = outer();
arrFn[0](); -> 1
arrFn[1](); -> 2
arrFn[2](); -> 3
```







push (x)
pop ()