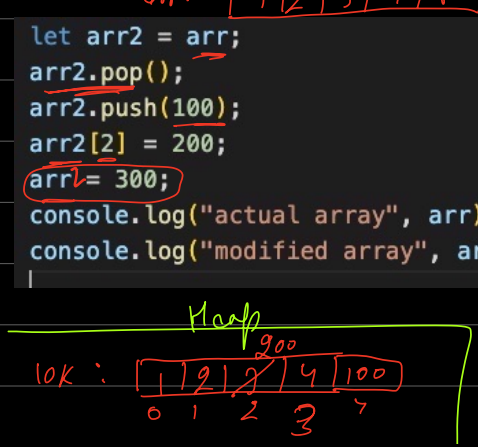
Agenda

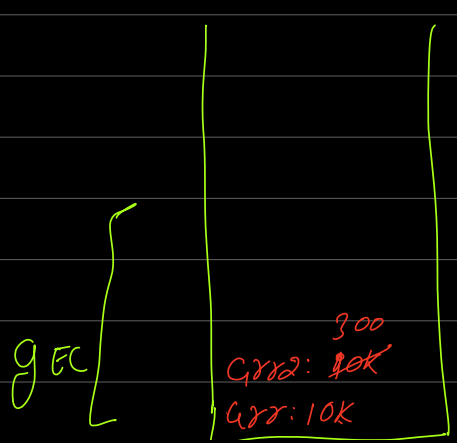
1. prototypal inheritance
2. spread, rest and default parameters
3. polyfill of call, bind, apply.
4. deep copy and shallow copy.
5. flatten an array.

(Array)$_{child}$ → [Array]$_{Parent}$ → [Object]$_{global}$ → null

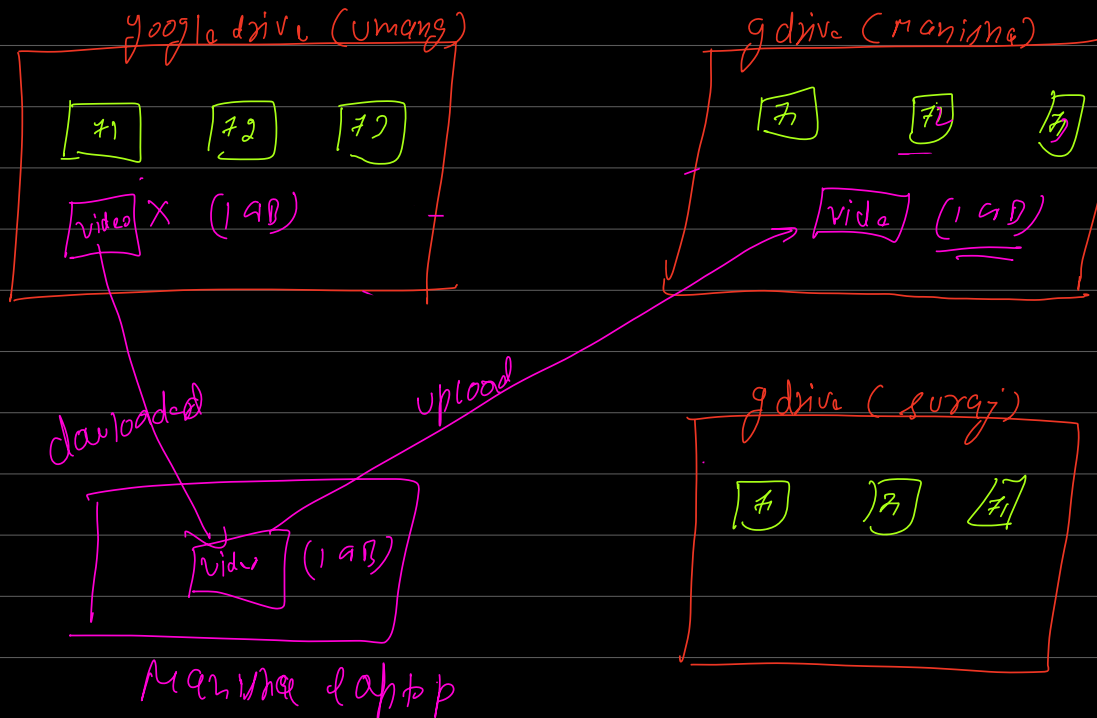
arr: | 1 | 2 | 3 | 4 | 5 |

```
let arr2 = arr;
arr2.pop();
arr2.push(100);
arr2[2] = 200;
arr = 300;
console.log("actual array", arr);
console.log("modified array", arr2);
```





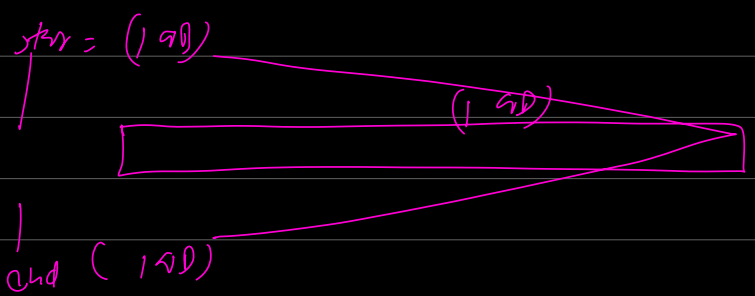
Stack



(3N)

arr: [1|2|3|4|5] (1GB) X

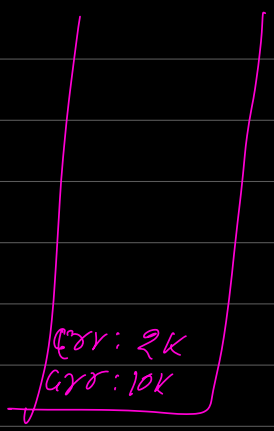
arr2: [1|2|3|4|5] (1GB) (2N)

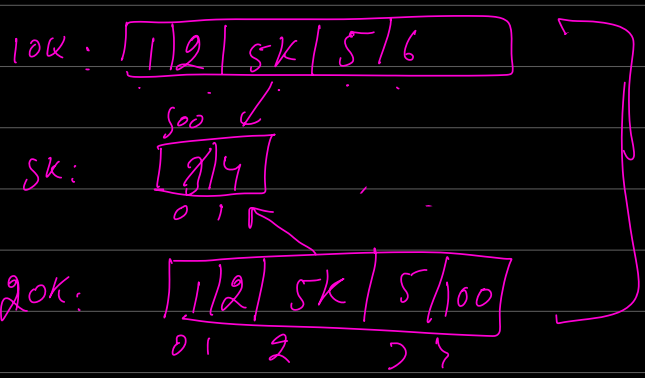


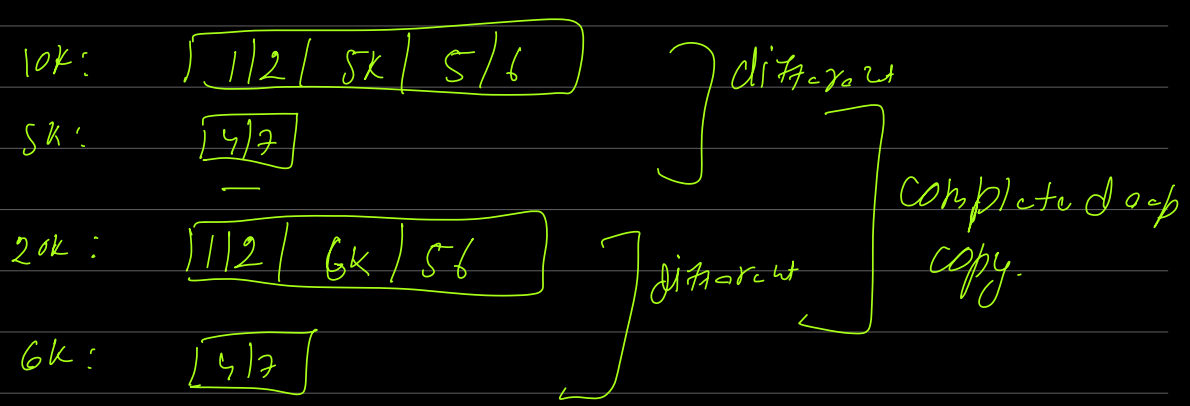
```
let arr = [1, 2, [3, 4], 5, 6];
// Spread copies value from one array to

let arr2 = [...arr];
arr2.pop();
arr2.push(100);
// arr2[2] = 300;
arr2[2][0] = 500;

console.log("actual Array: ", arr);
console.log("modified Array: ", arr2);
```







2D

arr:

1	2	3
4	5	6
7	8	9

} fake

r = arr.length

c = (arr[0].length)

arr: 0 → 10K, 1 → 20K, 2 → 30K

10K: 1 2 3

20K: 4 5 6

30K: 7 8 9

} r calcity

r = arr.length

c = arr[0].length

= (10K).length

= 3

3D

arr[N][M][2]

arr:

1	2	3
4	5	6
7	8	9

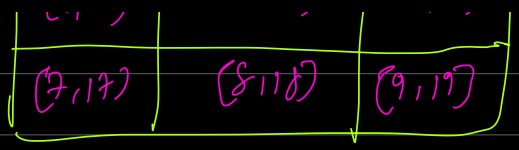
0

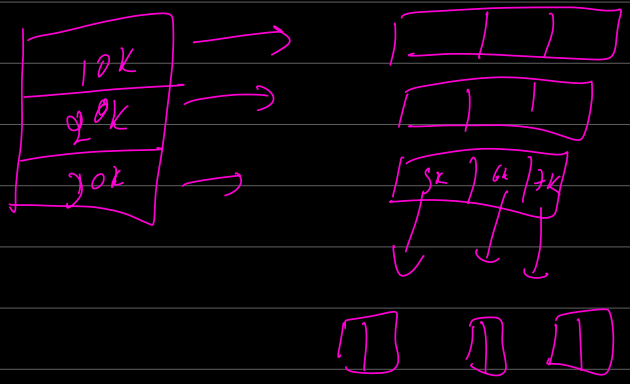
11	12	13
14	15	16
17	18	19

arr

0

[1,11]	[2,12]	[3,13]
[4,14]	[5,15]	[6,16]





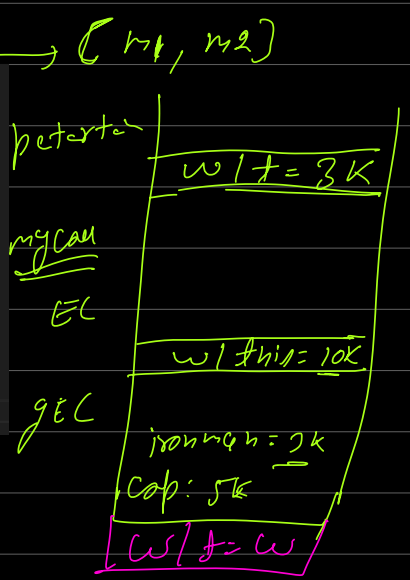
```
Function.prototype.myCall = function (requiredObject, ...arrayAsArgument) {
    // get your function.
    const functionToBeInvoked = this;

    // copy your function in the requiredObject for temp basis.
    requiredObject.tempFunction = functionToBeInvoked;

    // Call your function.
    requiredObject.tempFunction(...arrayAsArgument);

    // Delete temp method.
    delete requiredObject.tempFunction;
}

cap.petersTeam.myCall(ironMan, "thor", "loki",);
```



cap (- , - , petersteam : 10K)

iron (- , -)