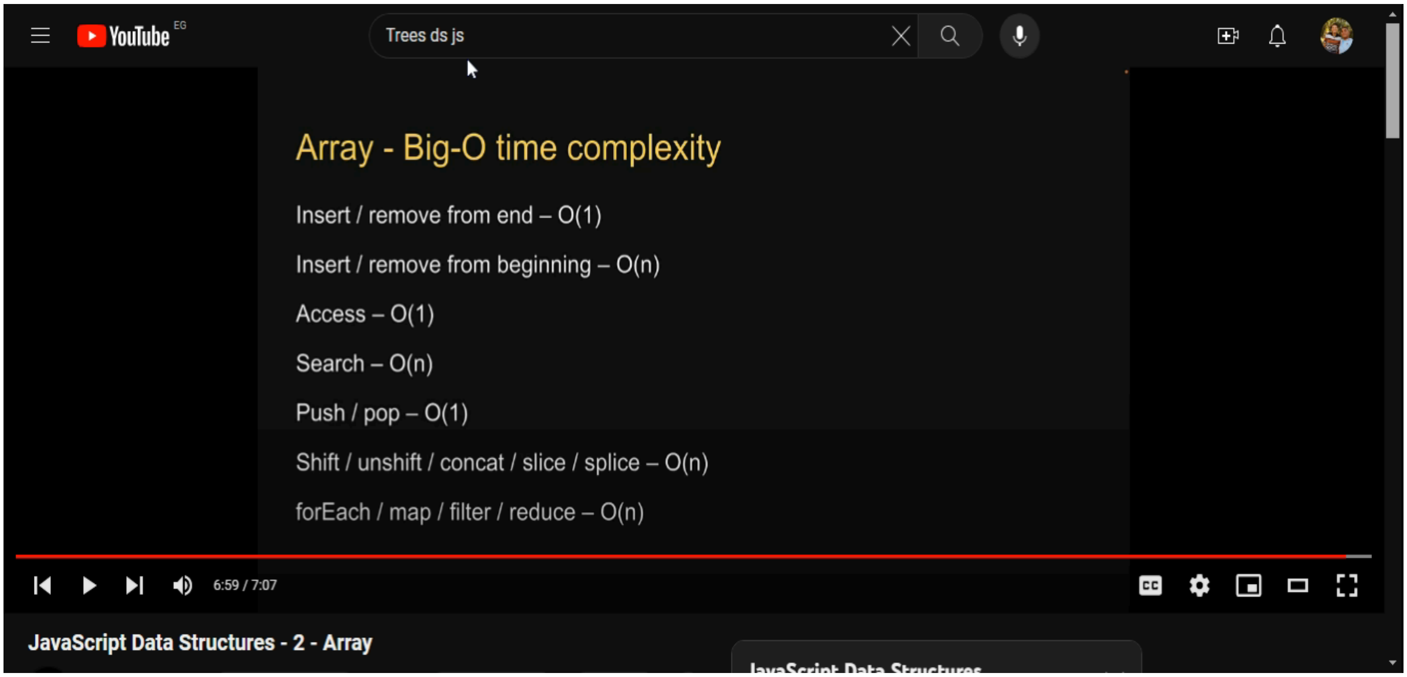
Data Structure using javascript

1)Data Structure is the way of store data so that it can be used efficiently

2)Array : -

1) is resizeable don’t have to declare size it’s dynamic
2) it iterable using for loop
3) it zero index based
4) it store mix of types [1,true,”Aaa”]
5) shift delete from begin unshift add to begin



YouTube EG
Trees ds js
6:59 / 7:07
JavaScript Data Structures - 2 - Array
JavaScript Data Structures

Array - Big-O time complexity

Insert / remove from end – O(1)

Insert / remove from beginning – O(n)

Access – O(1)

Search – O(n)

Push / pop – O(1)

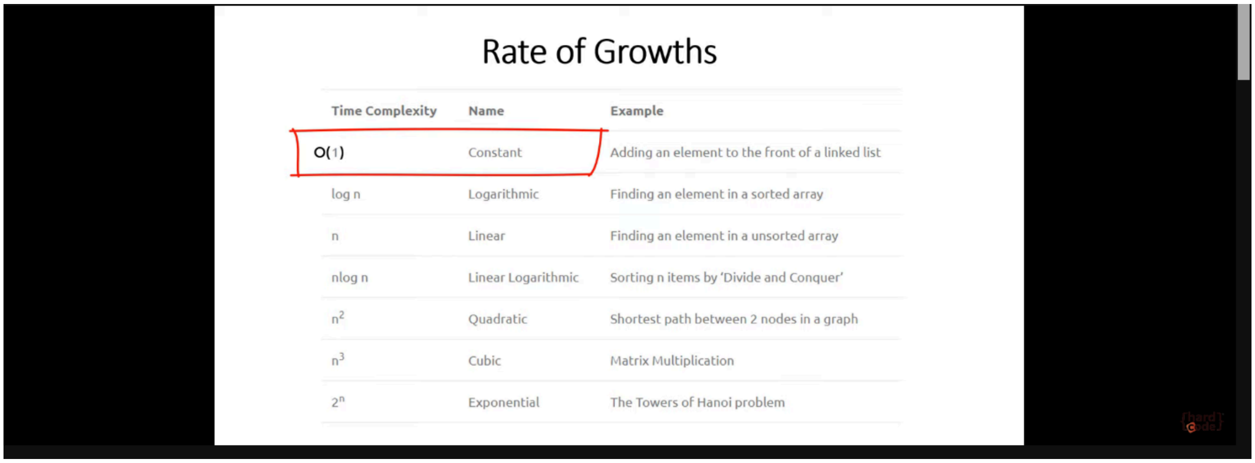
Shift / unshift / concat / slice / splice – O(n)

forEach / map / filter / reduce – O(n)

3) Asymptotic notation → delete non important things → 6n → o(n)

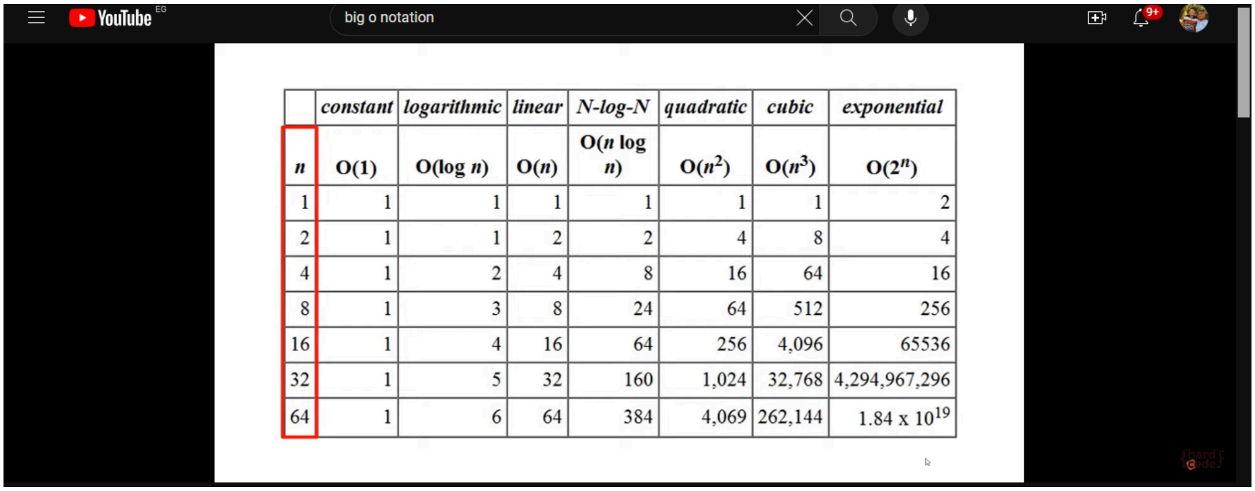
4) Big o notation → take the big order 2n→ O(n)

5) Big o notation → worst case scenario



Rate of Growths

Time Complexity	Name	Example
O(1)	Constant	Adding an element to the front of a linked list
log n	Logarithmic	Finding an element in a sorted array
n	Linear	Finding an element in a unsorted array
nlog n	Linear Logarithmic	Sorting n items by 'Divide and Conquer'
n^2	Quadratic	Shortest path between 2 nodes in a graph
n^3	Cubic	Matrix Multiplication
2^n	Exponential	The Towers of Hanoi problem

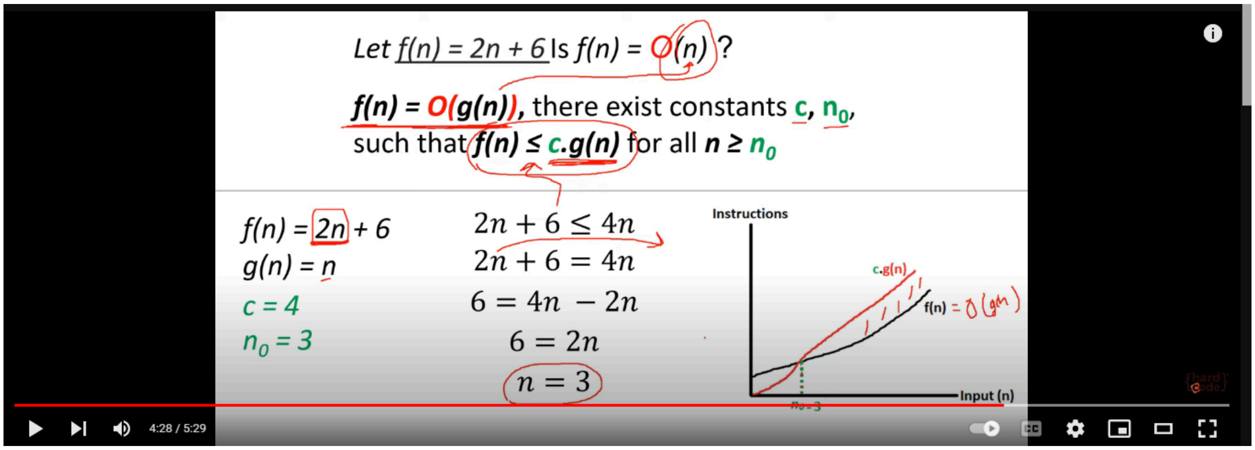


	constant	*logarithmic*	*linear*	*N-log-N*	*quadratic*	*cubic*	*exponential*
n	O(1)	O(log n)	O(n)	O(n log n)	O(n^2)	O(n^3)	O(2^n)
1	1	1	1	1	1	1	2
2	1	1	2	2	4	8	4
4	1	2	4	8	16	64	16
8	1	3	8	24	64	512	256
16	1	4	16	64	256	4,096	65536
32	1	5	32	160	1,024	32,768	4,294,967,296
64	1	6	64	384	4,069	262,144	1.84 x 10^{19}

6) Rate of growth → معدل الزيادة في كل خطوة

7) Big omega → best case

8) Big theta → average case



Let f(n) = 2n + 6 Is $f(n) = O(n)$?

$f(n) = O(g(n))$, there exist constants **c**, **n_0**, such that **$f(n) \leq c.g(n)$** for all **$n \geq n_0$**

$f(n) = \boxed{2n} + 6$

$g(n) = n$

$c = 4$

$n_0 = 3$

$$2n + 6 \leq 4n$$
$$2n + 6 = 4n$$
$$6 = 4n - 2n$$
$$6 = 2n$$
$$n = 3$$

9)