DSA

with **JAVASCRIPT**

Episode 1 - The Introduction and Strings

by ***Tech Jashwanth***

Topics Covered

1. *What is DSA?*
2. *Why DSA Matters for JS Developers*
3. *What is Time Complexity?*
4. *Big-O Notation (Most Common)*
5. *What is Space Complexity?*
6. *Strings in JavaScript*
7. *String Operation Time Complexities*
8. *Basic String Examples in JS*
9. *Next Episode Preview*
10. **50** *Practice Questions*

What is DSA?

Data Structures (DS)

Ways to store and organize data efficiently.

Algorithms (A)

Step-by-step methods to solve problems optimally.

Together:

DSA = Efficient storage + Efficient logic

Why DSA Matters for JS Developers?

without DSA

```
let numbers = [];
for (let i = 0; i < 500000; i++) {
  numbers.push(i);
}
let found = false;
for (let i = 0; i < numbers.length; i++) {
  if (numbers[i] === 499999) {  // last element
    found = true;
    break;
  }
}
console.log("Found:", found);
```

```
Slow Method: 62.47ms
Found: true
```

with DSA

```
let numberSet = new Set();
for (let i = 0; i < 500000; i++) {
  numberSet.add(i);
}

let foundFast = numberSet.has(499999);
console.log("Found:", foundFast);
```

```
Fast Method: 52.363ms
Found: true
```

Why DSA Matters for JS Developers?

- Helps write clean, optimized code
- Improves logic building & problem solving
- Makes large data processing faster
- Backbone of complex features (search, filters, chat)
- Important for interviews & coding rounds
- JS frameworks (React, Node, Next) internally use DS & Algorithms

What is Time Complexity?

Time Complexity = How the running time of your code grows as the input size increases.

It tells us:

1. How fast your logic scales
2. Whether code becomes slow on big data

We express it using Big-O Notation.

Big-O Notation

Big-O	***One-Line Description***
O(1)	Takes the same time no matter how big the input is.
O(log n)	Time reduces by half each step (very fast growth).
O(n)	Time grows directly with input size (one full loop).
O(n log n)	Grows slightly faster than linear (used in efficient sorting).
$O(n^2)$	Time grows very fast (nested loops).
$O(2^n)$	Doubles with every new input (very slow).
O(n!)	Grows extremely fast (worst possible).

Big O notations in order:

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) \rightarrow O(2^n) \rightarrow O(n!)$

What is Space Complexity?

Space Complexity = How much ***extra memory*** your algorithm uses.

Examples:

- Using only variables → O(1) space
- Creating a new array → O(n) space

Used to measure memory efficiency.

Strings

in JavaScript

Strings in JavaScript

- Strings are immutable
- Any modification creates a new string
- Stored as sequence of characters
- Used in 40% of beginner DSA problems
- Common operations:
  - Length
  - Access
  - Slice
  - Concat
  - Split
  - Search
  - Reverse

Strings

in Execution

String Operation Time Complexities

str='DSA Episode 1'

- str.length → **O(1)**
- str[i] → **O(1)**
- slice() → **O(n)**
- concat() → **O(n)**
- split() → **O(n)**
- reverse() → **O(n)**
- includes() / indexOf() → **O(n)**

Arrays

in Part 2

50

Practice Questions PDF is sent in
Telegram Channel

Telegram Channel Link

Youtube Channel Link