नमस्ते क्लास! आज का हमारा लेक्चर DSA (Data Structures and Algorithms) के एक बहुत ही महत्वपूर्ण और प्रैक्टिकल टॉपिक पर है: "Array Insertion in JavaScript".

पिछली क्लास में हमने Traversal सीखा था, यानी डेटा को देखना। आज हम सीखेंगे कि बने-बनाए Array में नया डेटा "घुसाना" (Insert करना) कैसे है। यह सुनने में आसान लगता है, लेकिन इसके पीछे की Memory Logic और Performance को समझना ही एक असली इंजीनियर की पहचान है।

Subject: DSA with JavaScript

Topic: Array Insertion (डेटा को जोड़ना और व्यवस्थित करना)

Level: Beginner to Intermediate | Language: Hindi (Hinglish Style)

1. Introduction (परिचय)

Array Insertion का मतलब है किसी मौजूदा Array में एक नया Element जोड़ना। यह काम तीन जगहों पर हो सकता है:

* At the End: सबसे पीछे जोड़ना।

* At the Beginning: सबसे आगे जोड़ना।

* At a Specific Index: बीच में कहीं भी अपनी मर्जी की जगह पर जोड़ना।

JavaScript में हमारे पास push() और unshift() जैसे रेडीमेड तरीके हैं, लेकिन DSA में हमें यह समझना पड़ता है कि अगर ये मेथड्स न हों, तो कंप्यूटर के अंदर "Shifting" कैसे होती है।

2. Why this topic is important? (यह क्यों ज़रूरी है?)

* Dynamic Data: सोचिए एक "Waitlist" है। जैसे ही नया यूजर आता है, उसे लिस्ट में जोड़ना पड़ता है।

* Undo/Redo Logic: हर नए एक्शन को हिस्ट्री वाले Array में इंसर्ट किया जाता है।

* Sorting Algorithms: कई सॉर्टिंग एल्गोरिदम (जैसे Insertion Sort) में डेटा को सही जगह "Insert" करना ही मुख्य लॉजिक होता है।

* Memory Management: आपको पता होना चाहिए कि डेटा डालने पर कंप्यूटर को कितनी मेहनत करनी पड़ रही है।

3. Core Concepts (गहराई से समझें)

A. End Insertion (O(1) Efficiency)

जब हम Array के अंत में कुछ जोड़ते हैं, तो बाकी एलिमेंट्स को अपनी जगह से हिलना नहीं पड़ता। कंप्यूटर बस देखता है कि अगली खाली जगह कहाँ है और वहाँ वैल्यू डाल देता है।

B. Beginning/Middle Insertion (O(n) Efficiency)

यह थोड़ा मुश्किल काम है। कल्पना कीजिए कि एक लाइन में 5 लोग खड़े हैं। अगर छठे इंसान को सबसे आगे (Index 0) खड़ा होना है, तो बाकी 5 लोगों को एक-एक कदम पीछे हटना पड़ेगा।

* Shifting: इंडेक्स 0 को खाली करने के लिए हमें बाकी सबको i+1 पोजीशन पर खिसकाना पड़ता है। इसमें समय ज़्यादा लगता है।

4. Code Examples & Explanation (कोडिंग और लॉजिक)

1. Insertion at the End (The Easy Way)

```
let students = ["Amit", "Rahul", "Priya"];
```

```
// Method 1: Built-in push()
students.push("Sneha");

// Method 2: Logic without push()
students[students.length] = "Sumit";

console.log(students); // ["Amit", "Rahul", "Priya", "Sneha", "Sumit"]
```

Logic: students.length हमेशा अगले खाली इंडेक्स का नंबर बताता है।

2. Insertion at a Specific Index (The Manual Logic)

इंटरव्यू में अक्सर पूछा जाता है: "बिना splice() इस्तेमाल किए किसी भी इंडेक्स पर डेटा कैसे डालें?"

```
function insertAt(arr, index, newValue) {
    // 1. पीछे से लूप चलाओ ताकि डेटा ओवरराइट न हो
    for (let i = arr.length - 1; i >= index; i--) {
        arr[i + 1] = arr[i]; // हर एलिमेंट को एक कदम आगे खिसकाओ
    }

    // 2. अब जगह खाली हो गई है, नया वैल्यू डालो
    arr[index] = newValue;
    return arr;
}

let myArr = [10, 20, 40, 50];
insertAt(myArr, 2, 30); // Index 2 पर 30 डालो
console.log(myArr); // [10, 20, 30, 40, 50]
```

Explanation: - हमने पीछे से (Reverse) लूप चलाया।

* अगर हम आगे से लूप चलाते, तो 40 की जगह 30 आ जाता और 40 हमेशा के लिए खो जाता (Data Loss)। इसलिए "Shifting" हमेशा पीछे से शुरू की जाती है।

3. Insertion Using splice() (The Modern Way)

JavaScript का splice() मेथड बहुत ताकतवर है। यह एक साथ कई काम कर सकता है।