Day 12 – [4th July 2025]

TOPICS COVERED

Array Destructuring:

Array destructuring allows us to unpack values from an array into individual variables.

Example:

```
const numbers = [10, 20, 30];
const [a, b, c] = numbers;
console.log(a); // 10
```

It helps avoid repetitive access like arr[0], arr[1], etc.

Object Destructuring:

Object destructuring lets us extract properties from an object into variables.

Example:

```
const user = { name: "Navpreet", age: 20 };
const { name, age } = user;
console.log(name); // "Navpreet"
```

This makes object handling cleaner and readable, especially in React props.

Spread (...) and Rest (...) Operators:

Both use the ... syntax but behave differently based on context:

Spread Operator – Expands values

Used to copy arrays/objects or pass elements as arguments.

```
const arr1 = [1, 2];
const arr2 = [...arr1, 3, 4]; // [1, 2, 3, 4]
const obj1 = { a: 1 };
const obj2 = { ...obj1, b: 2 }; // { a: 1, b: 2 }
```

Rest Operator – Gathers values:

Used to group remaining items into an array or object.

```
function sum(...nums) {
  return nums.reduce((a, b) => a + b);
}
console.log(sum(1, 2, 3)); // 6
const { a, ...rest } = { a: 1, b: 2, c: 3 };
console.log(rest); // { b: 2, c: 3 }
```

React Hooks – Introduction to useState:

In React, hooks are special functions that let us use React features (like state) in functional components.

Today I used the useState hook to store and update values dynamically in a React component.

Example:

```
import { useState } from 'react';
function App() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>Clicked: {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click Me</button>
    </div>
  );
}
```

This replaces the need for class-based state management and makes the code simpler and cleaner.

TOOLS USED:

Visual Studio Code

Node.js + npm

Vite + React

Chrome DevTools Console