🚀 JavaScript Day 8 Notes – Arrays and Their Methods

💡 1. Introduction to Arrays

An **Array** in JavaScript is a collection of items stored at **contiguous memory locations**. They can store multiple values in a **single variable**.

🔑 Creating Arrays

```
const myArr = [0, 1, 2, 3, 4, 5];

const myHero = ["IronMan", "Batman"];

const myArr2 = new Array(1, 2, 3, 4); // Using Array constructor
```

📝 Accessing Array Elements

```
console.log(myHero[1]); // Batman

console.log(myArr2[1]); // 2
```

🛠️ 2. Array Methods

📥 Adding and Removing Elements

- push(): Adds elements to the **end** of the array
- pop(): Removes the **last** element
- unshift(): Adds elements to the **start** of the array
- shift(): Removes the **first** element

✅ Example:

```
myArr.push(6); // Adds 6 to the end

myArr.pop();  // Removes the last element

myArr.unshift(7); // Adds 7 to the start

myArr.shift();   // Removes the first element
```

🔎 Searching in Arrays

- includes(value): Checks if the value exists in the array
- indexOf(value): Returns the **index** of the value (or -1 if not found)

📝 Example:

```
console.log(myArr.includes(9)); // false

console.log(myArr.indexOf(3));  // 3
```

🧩 Joining Elements

- join(): Joins all elements into a **single string**

✂️ Slice and Splice

- slice(start, end): Extracts a portion of the array **without modifying the original array**
- splice(start, deleteCount): Changes the array by removing/replacing items

📝 Example:

```
const myn1 = myArr.slice(1, 3); // Doesn't change original array

const myn2 = myArr.splice(1, 3); // Modifies original array
```

🦸 Combining Arrays

🪄 Concatenation

- concat(): Combines two arrays
- spread operator (...): Combines arrays more flexibly

📝 Example:

```
const marvel = ["thor", "Ironman", "Spiderman"];

const dc = ["superman", "batman", "flash"];

const heroes = [...marvel, ...dc];

console.log(heroes);
```

🏞️ Advanced Array Methods

- flat(): Flattens a nested array
- Array.isArray(): Checks if a variable is an array
- Array.from(): Creates an array from a string or iterable
- Array.of(): Creates an array from given arguments

📝 Example:

```
const another_array = [1, 2, 3, [4, 5, 6], 7, [8, 9, [2, 1]]];

const new_array = another_array.flat(Infinity);

console.log(new_array); // Flattens all nested arrays


console.log(Array.from("pranay")); // ["p", "r", "a", "n", "a", "y"]
console.log(Array.of(100, 200, 300)); // [100, 200, 300]
```