Array in Javascript 2

1. Recap & Array Basics

- The video briefly revisits arrays as **ordered collections** holding multiple values, indexed from 0.

2. Common Array Methods Covered

`.push()` and `.pop()`

- `push(item)` adds an element at the end.
- `pop()` removes the last element and returns it.
- Example:

```
let fruits = ['apple', 'banana'];
fruits.push('orange'); // ['apple', 'banana', 'orange']
let removed = fruits.pop(); // removed = 'orange', fruits = ['apple', 'banan
a']
```

`.shift()` and `.unshift()`

- `shift()` removes the first element and returns it.
- `unshift(item)` adds an element to the beginning.
- Example:

```
let nums = [1, 2, 3];
nums.shift(); // returns 1, nums = [2, 3]
nums.unshift(0); // nums = [0, 2, 3]
```

`.indexOf()` and `.includes()`

- `indexOf(item)` gives the index of the first match or –1 if not found.
- `includes(item)` gives `true` or `false` if item exists.

- Demo:

```
let colors = ['red', 'green', 'blue'];
colors.indexOf('green'); // 1
colors.includes('purple'); // false
```

`.splice()`

- Versatile method: can add, remove, or replace elements at any position.
- Usage:

```
let arr = [0,1,2,3,4];
arr.splice(2, 1, 'a', 'b');
// Removes 1 item at index 2, then inserts 'a' and 'b'
// arr = [0,1,'a','b',3,4]
```

`.slice()`

- Creates a new subarray from start to end (non-inclusive).
- Example:

```
let letters = ['a','b','c','d'];
let sub = letters.slice(1,3); // ['b','c']
```

`.concat()`

- Joins arrays and/or values into a new array.
- Example:

```
let a = [1,2];
let b = [3,4];
let c = a.concat(b, 5); // [1,2,3,4,5]
```

`.forEach()` and `.map()`

- `forEach(fn)` iterates through elements, running `fn` on each (no return).
- `map(fn)` transforms each element via `fn`, returning a new array.

```
[1,2,3].forEach(x ⇒ console.log(x));
let doubled = [1,2,3].map(x ⇒ x*2); // [2,4,6]
```

3. Real-Time Coding Demonstration

- The instructor works through real examples in Hindi again, showing how each method alters arrays inside the browser's console.

4. Good Practices

- She encourages always checking:
  - That `indexOf` result is not `1` before using it.
  - That array methods return expected types—like slice/map creating new arrays rather than modifying originals.
- Tips include using `let` vs `const` appropriately to prevent accidental changes.

📝 Quick-Look Notes for Revision

```
## JavaScript Array Methods – Chai Aur Code

- 📌 push(item): add at end
- 📌 pop(): remove last element
- 📌 unshift(item): add at start
- 📌 shift(): remove first element

- 📌 indexOf(item): returns index or -1
- 📌 includes(item): true if exists

- 📌 splice(start, deleteCount, ...items): insert/remove elements
- 📌 slice(start, end): subarray (non-destructive)
- 📌 concat(...arraysOrItems): merge into new array
```

- 📌 forEach(fn): iterate, no return
- 📌 map(fn): iterate & return transformed array

🔁 Always:
- Check `indexOf` != -1
- Understand mutate vs non-mutate behavior
- Use `const` when array reference doesn't change
