WEB APP DEVELOPMENT WITH REACTJS (INT252)

Lecture 2: Arrays, Objects & Immutability

Unit I – JavaScript Refresher & React Foundations

Syllabus Mapping:

- JavaScript arrays & objects
- ES6 spread operator
- Immutability principles in React

Target Learner: Beginner (HTML, CSS, JavaScript knowledge)

Why This Lecture Is Critical for React

React UI updates depend on:

- Arrays
- Objects
- Detecting changes

👉 If you mutate data, React may **not re-render**

JavaScript Arrays (Quick Refresh)

Arrays store multiple values:

```
const numbers = [1, 2, 3, 4];
```

In React, arrays are used to:

- Render lists
- Store state collections

The Most Important Array Method: `map()`

`map()`:

- Loops through array
- Returns a **new array**

```
const doubled = numbers.map(n => n * 2);
```

Why React Loves `map()`

React needs:

- Original data unchanged
- New data reference

`map()` does **not mutate** original array

Example: Rendering List Concept

```
const names = ["A", "B", "C"];

names.map(name => `<li>${name}</li>`);
```

This idea becomes JSX later

`filter()` – Selecting Items

```
const even = numbers.filter(n => n % 2 === 0);
```

Returns a **new array**

`reduce()` – Combining Values

```
const sum = numbers.reduce((total, n) => total + n, 0);
```

Used later for totals, counters

JavaScript Objects (Refresh)

```
const user = {
  name: "Aman",
  age: 20
};
```

Objects store structured data

Object Update (WRONG – Mutation)

```
user.age = 21; // ❌ mutates object
```

React cannot detect this change safely

Object Update (RIGHT – Immutability)

```
const updatedUser = {
  ...user,
  age: 21
};
```

Creates a **new object**

Spread Operator (`...`)

Spread:

- Copies values
- Works with arrays & objects

```
const arr2 = [...numbers];
```

Updating Arrays Immutably

❌ Wrong:

```
numbers.push(5);
```

✅ Right:

```
const newNumbers = [...numbers, 5];
```

Why Immutability Matters in React

React checks:

```
Old reference !== New reference
```

Only then it re-renders

Visual Diagram: Mutation vs Immutability

```
Mutation:
Same Object Reference ❌

Immutability:
New Object Reference ✅
```

Referential Equality (Simple Explanation)

```
const a = { x: 1 };
const b = a;

b.x = 2; // same reference
```

React thinks nothing changed

Creating New Reference (Correct)

```
const b = { ...a, x: 2 };
```

React detects change

React State Connection (Preview)

```
setItems([...items, newItem]);
```

This pattern appears everywhere

Common Beginner Mistakes

❌ Using push, pop, splice
❌ Direct object mutation
❌ Forgetting spread operator

Practice Exercises

1. Double array values using `map()`
2. Add item to array immutably
3. Update object property immutably

✅ Answers – Practice Exercises

1. map()

```
const result = [1,2,3].map(n => n * 2);
```

✅ Answers – Continued

2. Add Item

```
const arr = [1,2];
const newArr = [...arr, 3];
```

✅ Answers – Continued

3. Object Update

```
const user = { name: "Aman", age: 20 };
const updated = { ...user, age: 21 };
```

Key Takeaways

- React depends on immutability
- Always create new arrays/objects
- `map`, `filter`, `reduce` are essential
- Spread operator is your best friend

Next Lecture

Lecture 3: Functional Programming Concepts for React

Unit I – JavaScript Refresher & React Foundations