# WEB APP DEVELOPMENT WITH REACTJS (INT252)

## Lecture 18: Performance Optimization & Memoization

**Unit V – State Management & Advanced React**

**Syllabus Mapping:**

- Performance optimization
- Memoization techniques
- React.memo, useMemo, useCallback

**Target Learner:** Beginner → Intermediate

# Why Performance Matters

React re-renders components when:

- State changes
- Props change

Too many re-renders = slow app ❌

## What is Re-rendering?

Re-rendering means:

- React calls the component function again
- JSX is recalculated

Not always bad, but sometimes unnecessary

# Problem Scenario

```
function App() {
  const [count, setCount] = useState(0);
  return <ExpensiveComponent />;
}
```

`ExpensiveComponent` re-renders every time ❌

## Solution: Memoization

Memoization means:

- Remembering previous result
- Skipping unnecessary work

## Tool 1: React.memo

`React.memo`:

- Prevents re-render
- If props have NOT changed

# React.memo Example

```
const Header = React.memo(function Header({ title }) {
  console.log('Rendered');
  return <h1>{title}</h1>;
});
```

## When React.memo Works

- Same props
- Same reference

## Tool 2: useMemo

`useMemo`:

- Memoizes a value
- Avoids expensive calculations

## useMemo Syntax

```
const value = useMemo(() => compute(), [deps]);
```

# useMemo Example

```
const result = useMemo(() => {
  return slowFunction(count);
}, [count]);
```

## When to Use useMemo

- Heavy calculations
- Derived values

## Tool 3: useCallback

`useCallback`:

- Memoizes a function
- Prevents new function creation

## useCallback Syntax

```
const handler = useCallback(() => {
  doSomething();
}, [deps]);
```

## useCallback Example

```
const handleClick = useCallback(() => {
  setCount(count + 1);
}, [count]);
```

# useMemo vs useCallback

| useMemo | useCallback |
| --- | --- |
| Memoize value | Memoize function |

## Common Beginner Mistakes

❌ Overusing memoization
❌ Using without understanding
❌ Missing dependencies

## When NOT to Optimize

- Small apps
- Simple components

Premature optimization is bad ❌

## Practice Exercises

1. Wrap component using React.memo
2. Memoize expensive calculation
3. Prevent function recreation

# ✅ Answers – Practice Exercises

```
const Comp = React.memo(Comp);
```

```
const val = useMemo(() => heavy(), []);
```

```
const fn = useCallback(() => {}, []);
```

## Key Takeaways

- Re-renders affect performance
- React.memo prevents re-renders
- useMemo and useCallback optimize work

# Next Lecture

## Lecture 19: Redux & Global State (Introduction)

### Unit V – State Management & Advanced React