WEB APP DEVELOPMENT WITH REACTJS (INT252)

Lecture 12: useEffect Hook & Component Lifecycle

Unit III – State & Hooks

Syllabus Mapping:

- useEffect Hook
- Component lifecycle concepts
- Side effects in React

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

Why useEffect Exists

So far, React code:

- Runs during rendering

But some tasks are NOT rendering:

- Fetching data
- Timers
- DOM interaction

These are called **Side Effects**.

What is a Side Effect?

A side effect is:

- Anything outside UI rendering

Examples:

- API calls
- setTimeout / setInterval
- Local storage

Problem Without useEffect

```
fetch('api/data'); // ❌ runs every render
```

Causes performance issues

Solution: useEffect Hook

`useEffect` lets React:

- Run side effects safely
- Control WHEN they run

Importing useEffect

```
import { useEffect } from 'react';
```

Basic useEffect Syntax

```
useEffect(() => {
  // side effect code
}, []);
```

Understanding the Dependency Array

```
[]  → run once (on mount)
[x] → run when x changes
(no array) → run every render
```

Example: Run Once (Component Mount)

```
useEffect(() => {
  console.log('Component loaded');
}, []);
```

Example: Run on State Change

```
useEffect(() => {
  console.log('Count changed');
}, [count]);
```

Visual Lifecycle Flow

```
Component Mount
      ↓
useEffect runs
      ↓
State change
      ↓
useEffect re-runs
```

Cleanup Function

Used to clean resources

```
useEffect(() => {
  return () => {
    console.log('Cleanup');
  };
}, []);
```

Cleanup Use Cases

- Stop timers
- Remove event listeners
- Cancel API calls

Timer Example

```
useEffect(() => {
  const timer = setInterval(() => {
    console.log('Tick');
  }, 1000);

  return () => clearInterval(timer);
}, []);
```

Common Beginner Mistakes

❌ Forgetting dependency array
❌ Infinite loops
❌ Updating state wrongly inside effect

Practice Exercises

1. Log message when component loads
2. Run effect when value changes
3. Add cleanup to timer

✅ Answers – Practice Exercises

1. Run Once

```
useEffect(() => {
  console.log('Mounted');
}, []);
```

✅ Answers – Continued

2. Run on Change

```
useEffect(() => {
  console.log(value);
}, [value]);
```

✅ Answers – Continued

3. Cleanup

```
useEffect(() => {
  const id = setInterval(() => {}, 1000);
  return () => clearInterval(id);
}, []);
```

Key Takeaways

- useEffect handles side effects
- Dependency array controls execution
- Cleanup prevents memory leaks

Next Lecture

Lecture 13: Routing in React (React Router)

Unit IV – Routing & Advanced Concepts