

# WEB APP DEVELOPMENT WITH REACTJS (INT252)

## Lecture 21: Debugging, Error Handling & Best Practices

Unit VI – Debugging, Testing & Deployment

Syllabus Mapping:

- Debugging React applications
- Error handling techniques
- Best practices in React development

Target Learner: Beginner → Intermediate

# Why Debugging Matters

Every React developer faces:

- Errors
- Warnings
- Unexpected UI behavior

Debugging = finding & fixing problems 

## Common React Errors

- Component not rendering
- State not updating
- Props undefined
- Infinite re-renders

# Using Browser Console

Most basic debugging tool:

```
console.log(value);
```

Use it to:

- Check state
- Check props

# React Developer Tools

Browser extension:

- Chrome / Firefox

Features:

- Inspect components
- View props & state
- Track re-renders

# How React DevTools Help

You can:

- See component tree
- Inspect hooks
- Modify state temporarily

Very powerful 

# Understanding Error Messages

React errors tell:

- File name
- Line number
- What went wrong

👉 Read them carefully

## Common Error Example

```
Cannot read property 'name' of undefined
```

Meaning:

- Object is undefined
- You accessed it too early

# Fixing Undefined Errors

Use conditional rendering:

```
{user && <h1>{user.name}</h1>}
```

# Error Boundaries

Error boundaries:

- Catch rendering errors
- Prevent app crash

Only class components !

# Error Boundary Example

```
class ErrorBoundary extends React.Component {
  state = { hasError: false };

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  render() {
    if (this.state.hasError) {
      return <h1>Something went wrong</h1>;
    }
    return this.props.children;
  }
}
```

# Using Error Boundary

```
<ErrorBoundary>
  <App />
</ErrorBoundary>
```

# Debugging State Issues

Checklist:

- Correct initial state?
- Correct dependency array?
- State mutation?

# Avoid Infinite Loops

Bad example:

```
useEffect(() => {
  setCount(count + 1);
});
```

## Fix Infinite Loops

```
useEffect(() => {
  setCount(c => c + 1);
}, []);
```

## Best Practices – Folder Structure

```
src/  
  |- components/  
  |- pages/  
  |- hooks/  
  |- features/
```

## Best Practices – Code

- Small components
- Meaningful names
- Reusable logic

## Best Practices – State

- Local first
- Context second
- Redux last

## Practice Exercises

1. Debug a state bug
2. Add console logs
3. Wrap App with ErrorBoundary



## Answers – Practice

1. Check state updates
2. Use console.log
3. Wrap root component

## Key Takeaways

- Debugging is a skill
- Use DevTools
- Follow best practices

## Next Lecture

Lecture 22: Testing Basics in React

Unit VI – Debugging, Testing & Deployment