# Week 11 React Classroom Session

# Agenda

- Component Life Cycle
- Higher Order Component
- Error Boundary
- Advanced Hooks
- Context

# Component Life Cycle

- Every component in react has a life cycle and each life cycle has 3 phases.
- Three Phases in Component Life cycle
  - Mounting phase
  - Update Phase
  - UnMount Phase
- These phases help you to take control of the application when they are rendered or re rendered

# Higher Order Component HOC

- A higher-order component is a function that takes a component and returns a new component.
- Used when two or more components have identical implementation and the data passed to them are different, then we use HOC.
- Note that a HOC doesn’t modify the input component, nor does it use inheritance to copy its behavior. Rather, a HOC composes the original component by wrapping it in a container component. A HOC is a pure function with zero side-effects.

# Error Boundary

- Error Boundary is used to handle errors effectively so that the application users do not see huge set of errors on the screen when error is occured.
- We use componentDidCatch lifecycle method to catch such errors in react and show what the error in a simple Words like Page Failed to Load. Page Not available, User blocked message, etc.,

Refer for more here

# Advanced Hooks

- useCallBack hook is used to boost performance of the application by not re defining the functions until there is a change in them.
- memo is used to avoid re rendering the child components if the state or props are not changed.

- useReducer is usually preferable to useState when you have complex state logic that involves multiple sub-values or when the next state depends on the previous one.
- useReducer also lets you optimize performance for components that trigger deep updates because you can pass dispatch down instead of callbacks.

# Context

- Context is used to pass data to the components without using the tree structure of component flow.
- When the data in the context is updated, all the components which render the data will also get refreshed/re rendered.
- Context is primarily used when some data needs to be accessible by many components at different nesting levels. Apply it sparingly because it makes component reuse more difficult.

We need to create context first as shown below.
const MyContext = React.createContext(defaultValue);
We need MyContext.provider and MyContext .consumer to pass data to other components
Reference for more here

# Summary

# Thank You