React Lifecycle & Hooks

Class Component Lifecycle

• Mounting: constructor → render → componentDidMount
• Updating: render → componentDidUpdate
• Unmounting: componentWillUnmount

```
class MyComp extends React.Component {
  componentDidMount(){ console.log("Mounted"); }
  componentDidUpdate(){ console.log("Updated"); }
  componentWillUnmount(){ console.log("Unmounted"); }
  render(){ return <div>Hello</div>; }
}
```

Functional Component Lifecycle (Hooks)

• Achieved via useEffect, useLayoutEffect, useRef, etc.

useState

• Add state to functional components.

```
const [count, setCount] = useState(0);
<button onClick={()=>setCount(c=>c+1)}>{count}</button>
```

useEffect

• Runs side effects after render; cleanup for unmount.

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

useLayoutEffect

• Like useEffect but runs synchronously after DOM mutations, before paint.

```
useLayoutEffect(()=>{ console.log("Layout effect"); });
```

useRef

• Stores mutable values across renders; also access DOM nodes.

```
const inputRef = useRef(null);
<input ref={inputRef} />
```

useImperativeHandle (+ forwardRef)

• Expose custom methods to parent when using refs.

```
const Child = forwardRef((props, ref)=>{
  useImperativeHandle(ref, ()=>({ focus:()=>inputRef.current.focus()
}));
  return <input ref={inputRef} />;
});
```

forwardRef

• Pass parent's ref to child DOM node.

```
const Fancy = forwardRef((props, ref)=> <input ref={ref} {...props}
/>);
```

useMemo

• Memoize expensive values; recompute only if deps change.

```
const result = useMemo(()=>expensiveCalc(num), [num]);
```

useCallback

• Memoize function identity; prevents re-creations on re-render.

```
const handleClick = useCallback(()=>doSomething(id), [id]);
```

useId

• Generate unique, stable IDs for accessibility/labels.

```
const id = useId();
<label htmlFor={id}>Name</label>
<input id={id} />
```

useTranslation (i18n)

• Hook from react-i18next for multi-language apps.

```
const { t } = useTranslation();
<p>{t("welcome")}</p>
```

useTransition

• Mark state updates as non-urgent to keep UI responsive.

```
const [isPending, startTransition] = useTransition();
startTransition(()=> setFilter(newValue));
```