React Hooks — Lesson Notes

What are Hooks?

Hooks let you use state and other React features in function components. They were introduced in React 16.8.

useState

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

function Counter() {
  const [count, setCount] = useState(0);
  return <button onClick={() => setCount(count + 1)}>Count:
        {count}</button>;
}
```

useEffect

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

useEffect(() => {
  // Runs after every render
  document.title = `Count: ${count}`;
}, [count]); // Only re-run when count changes

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

useRef

```
const inputRef = useRef(null);
// Focus the input: inputRef.current.focus();
return <input ref={inputRef} />;
```

Custom Hooks

```
function useLocalStorage(key, initial) {
  const [value, setValue] = useState(
    () => JSON.parse(localStorage.getItem(key)) || initial
  );
  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);
  return [value, setValue];
}
```

Rules of Hooks

1. Only call hooks at the top level — never inside loops, conditions, or nested functions.
2. Only call hooks from React function components or custom hooks.

Key Takeaways

1. `useState` for local component state.
2. `useEffect` for side effects (API calls, subscriptions, DOM updates).
3. Always specify a dependency array in `useEffect` to avoid infinite loops.
4. Extract reusable logic into custom hooks (`useXxx`).
