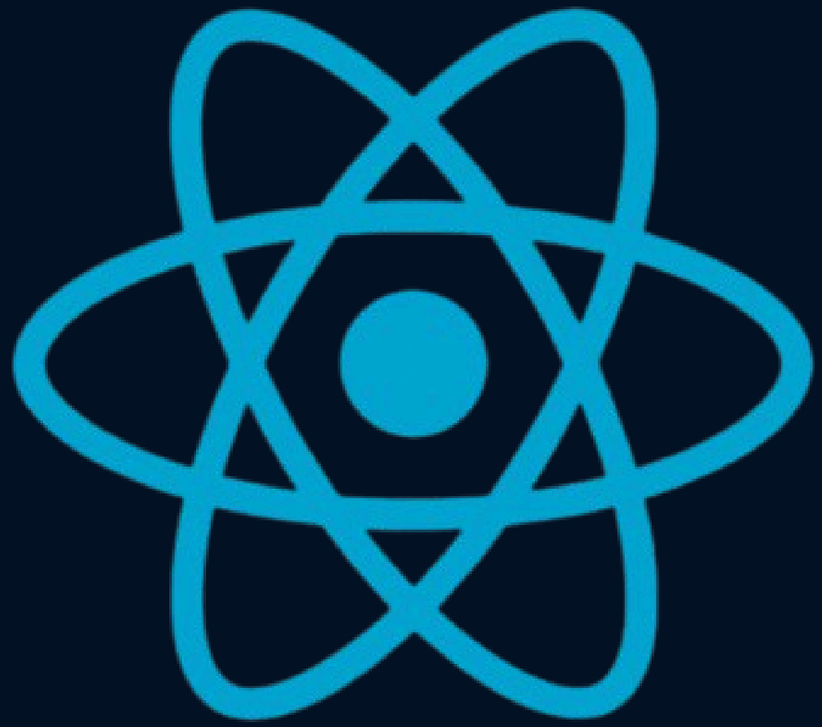
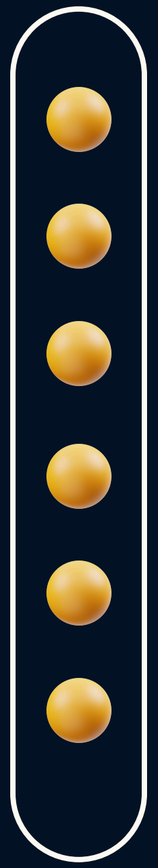
Otaku Tube

REACT JS NOTES (DAY 4)

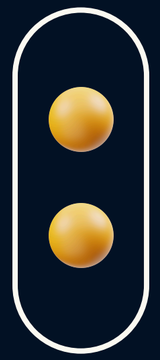


Present by Sewak Dhakal

TOPICS FOR DAY 4



- Objects and State
- Arrays and State
- How to Update an Object in State (Important!)
- Adding to an Array
- Removing from an Array
- Updating an Item in an Array



- Objects Inside Arrays in State
- Golden Rules for State

OBJECTS IN STATE

A state object is like a box with multiple labeled drawers.
Example:

```
const [person, setPerson] = useState({
  name: "Madara",
  age: 20,
});
```

Here, person is one state variable with two pieces of info.

HOW TO UPDATE AN OBJECT IN STATE (IMPORTANT!)

If you update only one property, you must keep the others too — React does not merge automatically like in class components.

❌ **Wrong (loses the other keys):**

```
setPerson({ name: "Sewak" });
```

Now age is gone.

✅ **Right (spread operator):**

```
setPerson({ ...person, name: "Sewak" });
```

This says: “Keep everything in person the same, but change the name.”

EXAMPLE:

```
export default function App() {
  const [person, setPerson] = useState({ name: "Madara", age: 20 });

  return (
    <div>
      <p>{person.name} is {person.age} years old.</p>
      <button onClick={() => setPerson({ ...person, age: person.age + 1 })}>
        Birthday!
      </button>
    </div>
  );
}
```

ARRAYS IN STATE

An array in state is like a list of things the robot is holding.
Example:

```
const [fruits, setFruits] = useState(["apple", "banana"]);
```

ADDING TO AN ARRAY

React state is immutable — you never directly change it (push, pop is bad). Instead, make a new array.

✅ **Add an item:**

```
setFruits([...fruits, "mango"]);
```

...fruits → copy old items
"mango" → add new one

REMOVING FROM AN ARRAY

✅ **Remove “banana”:**

```
setFruits(fruits.filter(fruit => fruit !== "banana"));
```

UPDATING AN ITEM IN AN ARRAY

✅ **Change “apple” to “grape”:**

```
setFruits(fruits.map(fruit => fruit === "apple" ? "grape" : fruit));
```

OBJECTS INSIDE ARRAYS IN STATE

Sometimes, you'll have lists of objects — like a todo list.
Example:

```
const [todos, setTodos] = useState([
  { id: 1, task: "Code", done: false },
  { id: 2, task: "Read", done: false }
]);
```

ADD A NEW TODO:

```
setTodos([...todos, { id: 3, task: "Sleep", done: false }]);
```

UPDATE A TODO:

```
setTodos(
  todos.map(todo =>
    todo.id === 2 ? { ...todo, done: true } : todo
  )
);
```

REMOVE A TODO:

```
setTodos(todos.filter(todo => todo.id !== 1));
```

GOLDEN RULES FOR STATE

Never mutate directly → always create a new object/array when updating.

Spread (...) is your best friend for copying.

State updates are async → don’t expect console.log right after to show the new value immediately.

One state = one source of truth → don’t keep duplicate values in state if you can compute them from something else.