📌 React의 State (상태 관리)

🚀 1. State란?

📖 개념

- **State(상태)** 는 React 컴포넌트에서 변화하는 데이터를 관리하는 객체입니다.
- State가 변경되면 React는 **자동으로 해당 컴포넌트를 다시 렌더링하여 UI를 업데이트**합니다.
- **컴포넌트 내부에서만 관리할 수 있으며, 부모-자식 간 데이터 전달에는 props를 사용해야 합니다.**

✅ State의 주요 특징

1. **컴포넌트의 동적인 데이터 관리** → 사용자의 입력, API 응답, UI 변경 등을 저장 가능
2. **불변성 유지 필요** → 직접 변경하지 않고, `setState` 또는 `useState`를 통해 상태 변경
3. **State 변경 시 렌더링 발생** → React는 변경된 부분만 효율적으로 다시 렌더링

💡 2. State 사용법

📌 1) `useState`를 활용한 기본적인 상태 관리

React의 함수형 컴포넌트에서는 **`useState` 훅을 사용하여 상태를 관리**할 수 있습니다.

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

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>현재 카운트: {count}</p>
      <button onClick={() => setCount(count + 1)}>증가</button>
    </div>
  );
}
```

✅ `useState(초기값)`을 사용하면 상태 변수(`count`)와 상태 변경 함수(`setCount`)가 반환됨

✅ `setCount(newValue)`를 호출하면 `count`가 변경되며, 컴포넌트가 다시 렌더링됨

🔥 3. State의 불변성 유지 (Immutable State)

📌 1) 상태를 직접 변경하면 안 되는 이유

```
const [user, setUser] = useState({ name: "Alice", age: 25 });

// ❌ 잘못된 방법 (직접 변경)
user.age = 30;
```

✅ **React는 상태가 직접 변경된 것을 감지하지 못하여, UI가 업데이트되지 않음**

📌 2) 올바른 상태 변경 방법 (새로운 객체 반환)

```
setUser({ ...user, age: 30 });
```

✅ **Spread 연산자(`...`)를 사용하여 기존 객체를 복사한 후, 변경된 값만 덮어쓰기**

🛠 4. State 업데이트 시 주의할 점

📌 1) 상태 업데이트는 비동기적으로 처리됨

```
const [count, setCount] = useState(0);

function handleClick() {
  setCount(count + 1);
  setCount(count + 1);
  setCount(count + 1);
}
```

✅ 예상값: +3, 실제값: +1 → React가 여러 개의 `setState` 호출을 **배치 처리(batch update)** 하기 때문

📌 2) 이전 상태 값을 기반으로 업데이트하려면 함수형 업데이트 사용

```
setCount(prevCount => prevCount + 1);
setCount(prevCount => prevCount + 1);
setCount(prevCount => prevCount + 1);
```

✅ `prevCount`를 사용하여 이전 상태 값을 안전하게 업데이트 가능

🎯 5. 여러 개의 State 관리

📌 1) 여러 개의 `useState` 사용

```
const [name, setName] = useState("Alice");
const [age, setAge] = useState(25);
```

✅ 작은 단위로 나누면 상태 관리를 더욱 효율적으로 할 수 있음

📌 2) 객체로 State 관리

```
const [user, setUser] = useState({ name: "Alice", age: 25 });

function updateAge() {
  setUser(prevUser => ({ ...prevUser, age: prevUser.age + 1 }));
}
```

✅ 객체를 상태로 관리하면 연관된 데이터를 함께 저장할 수 있음

🏁 6. 결론

✅ State의 핵심 개념 정리

1. State는 React에서 동적인 UI를 관리하는 핵심 요소
2. `useState`를 사용하여 상태를 정의하고, `setState`를 통해 업데이트 가능
3. 상태 변경 시 불변성을 유지해야 React가 올바르게 렌더링을 수행함
4. 여러 개의 State를 효율적으로 관리하려면 `useState`를 분리하거나, 객체 형태로 저장 가능
5. State 변경은 비동기적으로 처리되며, 이전 상태 값을 기반으로 업데이트할 경우 콜백 함수를 활용해야 함

✅ 결론

- State는 React에서 UI를 동적으로 변경하는 핵심 개념이며, 올바른 관리 방법을 익히는 것이 중요
- 불변성을 유지하면서 효율적인 상태 관리를 하면 성능 최적화와 유지보수성이 향상됨