3. useEffect로 라이프사이클 제어하기

다중 선택	라이프사이클

- **빈 배열 (`[]`)**
  - 이펙트는 한 번만 실행됩니다. (마운트 시)
  - `componentDidMount`와 유사한 동작.
- **배열 생략**
  - 상태/props가 변경될 때마다 실행됩니다.
  - `componentDidUpdate`와 유사.
- **특정 값 포함 (`[value]`)**
  - `value`가 변경될 때만 이펙트가 실행됩니다.
  - 의존성 관리가 중요.

예시

```
// src/App.jsx

import "./App.css";
import Viewer from "./components/Viewer";
import Controller from "./components/Controller";
import { useState, useEffect, useRef } from "react";
import Even from "./components/Even";

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

  const isMount = useRef(false);

  // 1. 마운트 : 탄생
```

```
  useEffect(() => {
    console.log("mount");
  }, []);

  // 2. 업데이트 : 변화, 리렌더링
  useEffect(() => {
    // 마운트 될 때 말고 진짜 업데이트 하는 경우에만 실행시키고 싶은 경우(
    // 사용할 수 있는 플래그(이거 없으면 마운트 될 때도 이 코드 실행됨)
    if (isMount.current) {
      isMount.current = true;
      return;
    }
    console.log("update");
  });

  // 3. 언마운트 : 죽음
  useEffect(() => {
    console.log("mount");
  }, []);

  const onClickButton = (value) => {
    setCount(count + value);
  };
  return (
    <div className="App">
      <h1>Simple Counter</h1>
      <section>
        <Viewer count={count} />
        {/* count가 짝수일때만 Even 컴포넌트 렌더링 */}
        {count % 2 === 0 ? <Even /> : null}
      </section>
      <section>
        <Controller onClickButton={onClickButton} />
      </section>
    </div>
  );
}
```

```
export default App;
```

```
// src/components/Even.jsx

import { useEffect } from "react";

const Even = () => {
  useEffect(() => {
    // 클린업, 정리함수 : useEffect가 끝날 때 실행됨
    return () => {
      console.log("unmount");
    };
  }, []);

  return <div>짝수입니다.</div>;
};

export default Even;
```