useEffect

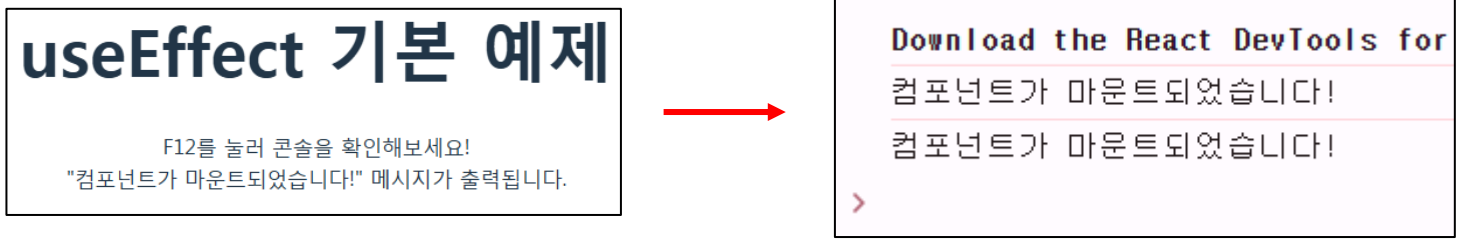
복습 문제

예제1] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 컴포넌트가 처음 렌더링될 때 콘솔에 메시지가 출력되도록 작성하시오.
② useEffect를 사용하여 마운트 시 "컴포넌트가 마운트되었습니다!" 출력 하시오.
③ 빈 의존성 배열 사용해 작성 하시오.
④ 함수 이름은 Ex01.jsx로 작성하시오.
⑤ src 폴더 안에 Effect 폴더 -> Ex01.jsx 파일을 작성하시오.
⑥ src/App.jsx에서 Ex01.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>



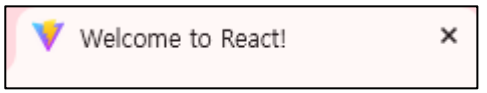
예제2] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 컴포넌트가 마운트될 때 브라우저 탭 제목을 변경되도록 작성하시오.
② useEffect로 document.title 변경되도록 작성하시오.
③ 마운트 시 "Welcome to React!"로 변경하시오.
④ 빈 의존성 배열 사용해 작성 하시오.
⑤ 함수 이름은 Ex02.jsx로 작성하시오.
⑥ src 폴더 안에 Effect 폴더 -> Ex02.jsx 파일을 작성하시오.
⑦ src/App.jsx에서 Ex03.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

Title Changer
브라우저 탭을 확인해보세요!
제목이 "Welcome to React!"로 변경되었습니다.

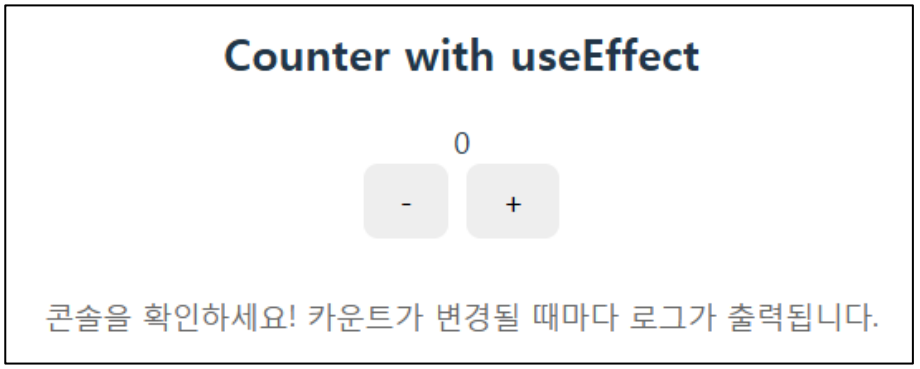


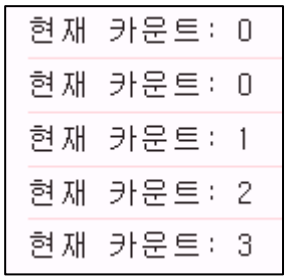
예제3] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 카운터 값이 변경될 때마다 콘솔에 현재 값이 출력되도록 작성하시오.
② useState로 카운터 구현하여 작성하시오.
③ useEffect로 count 값 변경 감지
④ count를 의존성 배열에 추가해 작성하시오.
⑤ 함수 이름은 Ex03.jsx로 작성하시오.
⑥ src 폴더 안에 Effect 폴더 -> Ex03.jsx 파일을 작성하시오.
⑦ src/App.jsx에서 Ex03jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>



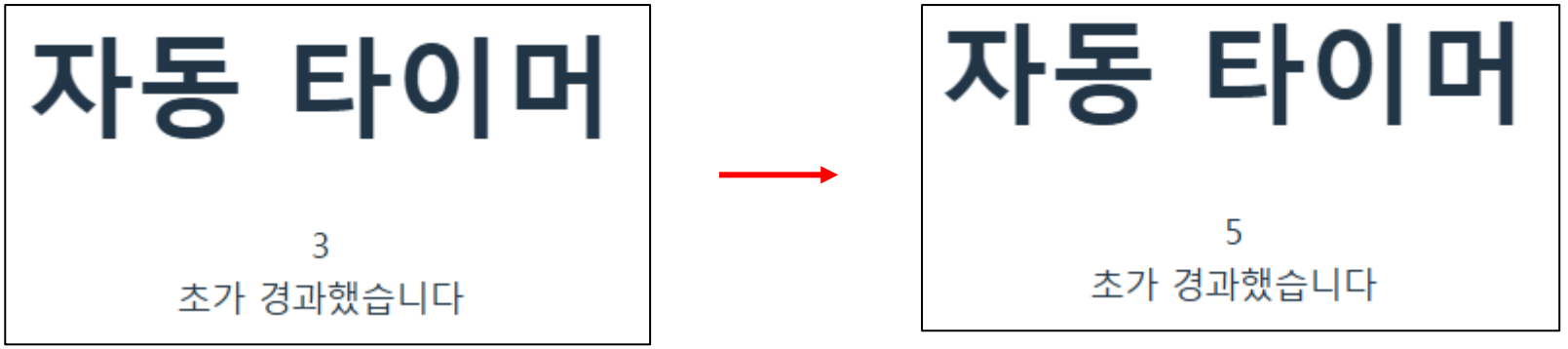


예제4] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 1초마다 자동으로 증가하는 타이머를 작성하시오.
② setInterval 사용해 작성하시오.
③ useEffect의 cleanup 함수로 타이머 정리하도록 작성하시오.
④ 의존성 배열에 추가해 작성하시오.
⑤ 함수 이름은 Ex04.jsx로 작성하시오.
⑥ src 폴더 안에 Effect 폴더 -> Ex04.jsx 파일을 작성하시오.
⑦ src/App.jsx에서 Ex04.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

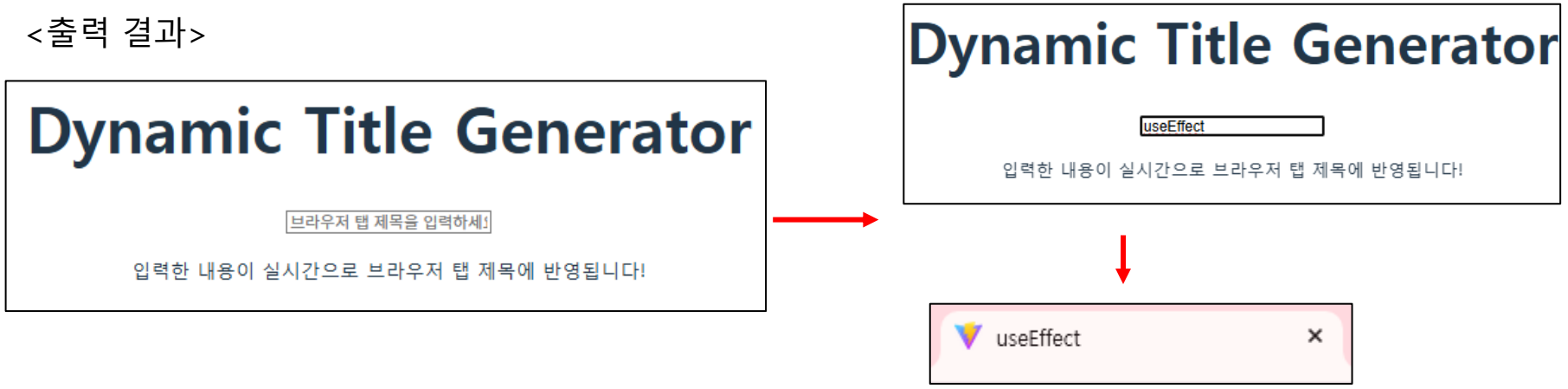


예제5] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 입력 필드의 값이 변경될 때마다 브라우저 탭 제목이 업데이트 되도록 작성하시오.
② input으로 텍스트 입력받도록 작성하시오.
③ useEffect로 input 값 변경 감지
④ title를 의존성 배열에 추가해 작성하시오.
⑤ 함수 이름은 Ex05.jsx로 작성하시오.
⑥ src 폴더 안에 Effect 폴더 -> Ex05.jsx 파일을 작성하시오.
⑦ src/App.jsx에서 Ex05.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>



예제6] 다음의 조건에 만족하도록 React를 작성 하시오.

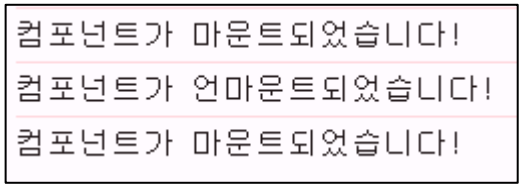
조건

① 컴포넌트가 나타나고 사라질 때 메시지를 출력하는 토글 가능한 컴포넌트를 작성하시오.
② 토글 버튼으로 자식 컴포넌트 표시/숨김 작성하시오.
③ 마운트 시 콘솔에 "컴포넌트 마운트" 출력되도록 작성하시오.
④ 언마운트 시 cleanup에서 "컴포넌트 언마운트" 출력되도록 작성하시오.
⑤ 부모 컴포넌트 : Ex06, 자식 컴포넌트 : ChildEx06으로 작성하시오.
⑥ 자식 컴포넌트 ChildEx06에 useEffect를 작성하시오.
⑦ 함수 이름은 Ex06.jsx로 작성하시오.
⑧ src 폴더 안에 Effect 폴더 -> Ex06.jsx 파일을 작성하시오.
⑨ src/App.jsx에서 Ex06.jsx을 import하여 실행 결과를 확인하시오.

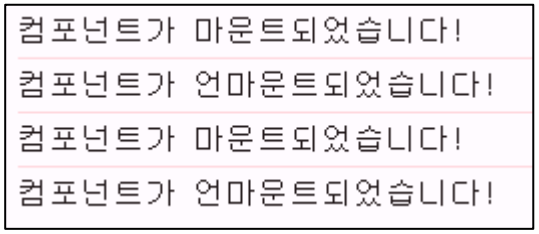
<출력 결과>


Lifecycle Demo
컴포넌트 보기
F12를 눌러 콘솔을 확인하세요!
컴포넌트의 마운트/언마운트 메시지가 출력됩니다.


Lifecycle Demo
컴포넌트 숨기기
안녕하세요! 저는 토글 가능한 컴포넌트입니다!
F12를 눌러 콘솔을 확인하세요!
컴포넌트의 마운트/언마운트 메시지가 출력됩니다.


컴포넌트가 마운트되었습니다!
컴포넌트가 언마운트되었습니다!
컴포넌트가 마운트되었습니다!


Lifecycle Demo
컴포넌트 보기
F12를 눌러 콘솔을 확인하세요!
컴포넌트의 마운트/언마운트 메시지가 출력됩니다.


컴포넌트가 마운트되었습니다!
컴포넌트가 언마운트되었습니다!
컴포넌트가 마운트되었습니다!
컴포넌트가 언마운트되었습니다!

예제7] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 현재 시간을 1초마다 업데이트하는 디지털 시계를 만드세요.

② 시:분:초 형식으로 표시되도록 작성하시오.

③ setInterval로 1초마다 업데이트되도록 작성하시오.

④ cleanup으로 타이머 정리함수도 작성하시오.

⑤ new Date() 이용해 작성하시오.

⑥ currentTime.getHours(), currentTime.getMinutes(), currentTime.getSeconds() 이용해 작성하시오.

⑦ 시, 분, 초를 두 자리 숫자로 맞춰주는 함수도 생성하여 작성하세요.

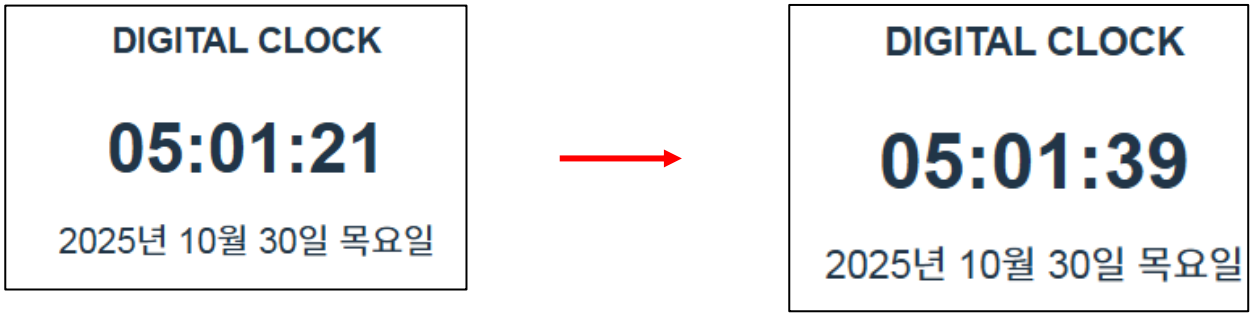
⑧ 날짜를 보기 좋은 형태로 포맷하는 currentTime.toLocaleDateString( )도 이용해 작성하시오.

⑨ 함수 이름은 Ex07.jsx로 작성하시오.

⑩ src 폴더 안에 Effect 폴더 -> Ex07.jsx 파일을 작성하시오.

⑪ src/App.jsx에서 Ex07.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>


DIGITAL CLOCK
05:01:21
2025년 10월 30일 목요일
DIGITAL CLOCK
05:01:39
2025년 10월 30일 목요일

예제8] **다음의 조건에 만족하도록 React를 작성 하시오.**

조건

① 두 개의 input 값을 곱한 결과를 계산하고, 변경될 때마다 콘솔에 출력 되도록 작성하시오.

② 두 개의 숫자 input 에서 입력받아 사용하세요

③ useEffect, useState이용해 작성하시오.

④ 의존성 배열 [num1, num2] 형식으로 다중 의존성 배열을 이용해 num1이나 num2가 바뀔 때만 실행되도록 작성하시오.

⑤ 곱셈 결과 표시 및 콘솔 출력되도록 작성하시오.

⑥ 함수 이름은 Ex08.jsx로 작성하시오.

⑦ src 폴더 안에 Effect 폴더 -> Ex08.jsx 파일을 작성하시오.

⑧ src/App.jsx에서 Ex08.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>


곱셈 계산기
첫 번째 숫자 0
두 번째 숫자 0
결과:
0
콘솔을 열면 계산 과정도 확인할 수 있습니다.
곱셈 계산기
첫 번째 숫자 3
두 번째 숫자 4
결과:
12
콘솔을 열면 계산 과정도 확인할 수 있습니다.
0 × 0 = 0
0 × 0 = 0
1 × 0 = 0
2 × 0 = 0
3 × 0 = 0
3 × 4 = 12

예제9] **다음의 조건에 만족하도록 React를 작성 하시오.**

조건

① 간단한 메시지를 입력하고 [전송] 버튼을 클릭하면 전송된 메시지는 의 에 추가되도록 자성하시오.

② 메시지를 입력 할 때마다 콘솔에도 현재 입력하는 메시가 출력되도록 작성하시오.

③ useEffect, useState이용해 작성하시오.

~~④ 3초간 입력이 없으면 표시 제거되도록 작성하시오.~~

~~⑤ title를 의존성 배열에 추가해 작성하시오.~~

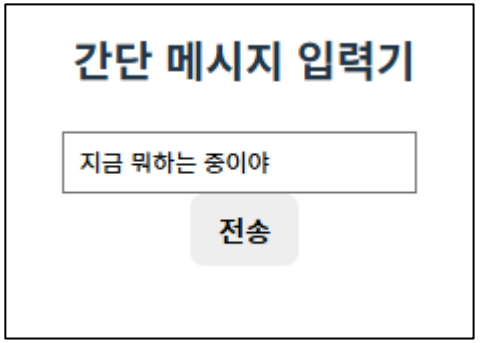
⑥ 함수 이름은 Ex09.jsx로 작성하시오.

⑦ src 폴더 안에 Effect 폴더 -> Ex09.jsx 파일을 작성하시오.

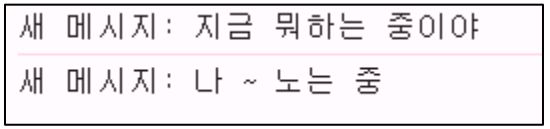
⑧ src/App.jsx에서 Ex09.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

<채팅 내용 입력 중 일 때>



<채팅 입력 완료 후>



예제10] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① 입력한 검색어로 **미리 만들어둔 사용자 배열**에서 필터링을 이용해 작성하시오.
② **useEffect**를 사용해서 검색어가 바뀔 때 결과 업데이트되도록 작성하시오.
③ 화면에는 **input**과 **검색 결과 목록**만 표시되도록 작성하시오.
④ useState이용해 작성하시오.
⑤ Includes()를 이용해 작성하시오.
⑥ 아래 오브젝트 배열을 이용해 작서하시오.

```
const users = [
        { id: 1, name: 'Alice', email: 'alice@example.com' },
        { id: 2, name: 'Bob', email: 'bob@example.com' },
       { id: 3, name: 'Charlie', email: 'charlie@example.com' },
        { id: 4, name: 'David', email: 'david@example.com' },
      ];
```

① 함수 이름은 Ex10.jsx로 작성하시오.
② src 폴더 안에 Effect 폴더 -> Ex10.jsx 파일을 작성하시오.
③ src/App.jsx에서 Ex10.jsx을 import하여 실행 결과를 확인하시오.

➢ Includes 란

- includes는 문자열이나 배열에서 부분 일치(contains) 를 검사할 때 사용 검색 기능에 자주 쓰인다.

1) 문자열에서 includes

```
// 대소문자 구분: 예시
'Hello'.includes('he')         // false
'Hello'.toLowerCase().includes('he')  // true
```

반환값: true 또는 false.
대소문자 구분(case-sensitive) 하므로 검색할 때 보통 .toLowerCase() 또는 .toUpperCase()로 맞춰서 비교한다.

2) 배열에서 includes

```
const arr = [1, 2, 3];
arr.includes(2)   // true
arr.includes(4)   // false
```

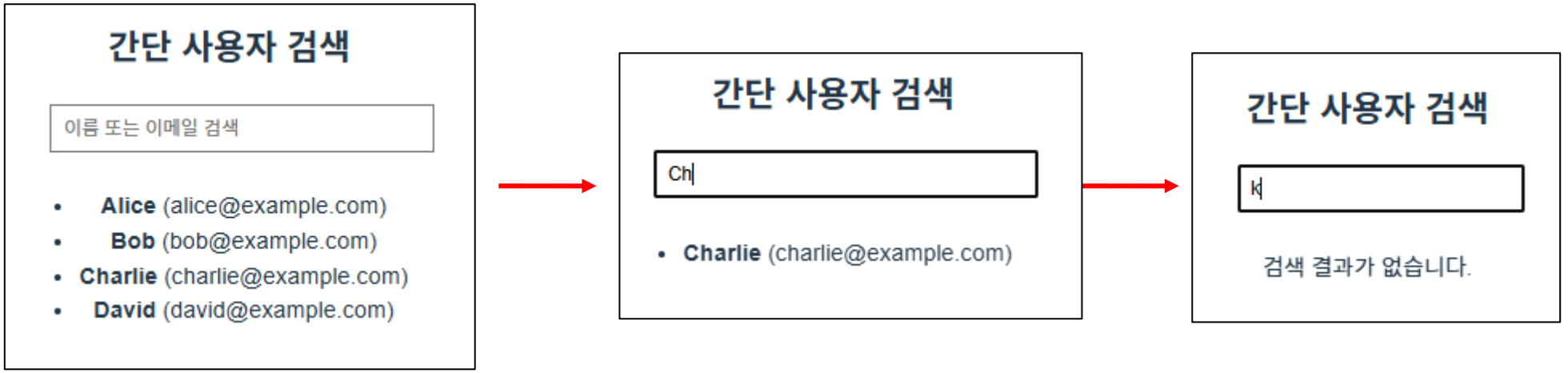
원시값(숫자, 문자열, 불리언)에 대해 동일성(strict equality, ===)으로 검색.
객체({})는 참조가 달라서 같은 구조라도 includes가 false가 될 수 있음:

3) 시작 위치 지정 (문자열의 두 번째 인자)

문자열 includes는 두 번째 인수로 시작 인덱스를 줄 수 있다.

```
'Javascript'.includes('a', 3)  // true (인덱스 3 이후에 'a'가 있음)
'Javascript'.includes('J', 1)  // false
```

<출력 결과>


간단 사용자 검색
이름 또는 이메일 검색
Alice (alice@example.com)
Bob (bob@example.com)
Charlie (charlie@example.com)
David (david@example.com)
간단 사용자 검색
Ch
Charlie (charlie@example.com)
간단 사용자 검색
k
검색 결과가 없습니다.