4강

데이터 흐름

➢ 컴포넌트 간 데이터 흐름과 상태 관리

1. 부모에서 자식으로 데이터 전달 (props)
2. 자식에서 부모로 이벤트 전달
3. 컴포넌트 간 데이터 흐름 이해

➢ Props 란

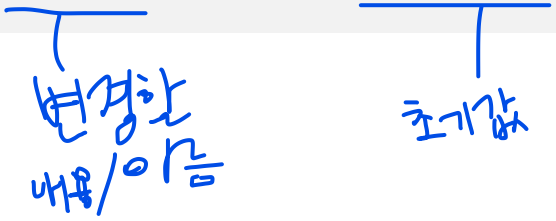
- props는 부모 → 자식으로 전달되는 읽기 전용 데이터 이다. (자식에서 직접 변경 불가하다)
- Props는 컴포넌트에서 컴포넌트로 전달하는 데이터를 말한다. 다시 말해 함수의 매개변수와 같은 개념이다.
- Props를 사용하면 컴포넌트를 효율적으로 재사용할 수 있다.
- 아래 Child 컴포넌트를 여러 번 재 사용해서 여러 명의 학생이름을 출력하였다.

```jsx
import Child from './child';
function Parent() {
  return (
    <>
      <Child name="홍길동" />
      <Child name="김영희" />
      <Child name="박철수" />
    </>
  );
}
export default Parent;
```

```jsx
function Child({ name }) {
  return (
    <>
      <h2>안녕하세요~ {name}님</h2>
    </>
  );
}
export default Child;
```

➢ useState 란

- 짧게 요약하면: useState는 React 함수형 컴포넌트에서 컴포넌트의 상태(state)를 선언하고 관리하게 해 주는 훅(hook)이다.
- 상태(state): UI에 영향을 주는 값(예: 입력값, 토글 여부, 카운터 등). 상태가 바뀌면 React가 컴포넌트를 다시 렌더링한다.
- useState는 상태 값과 그 값을 갱신하는 함수(업데이터)를 반환한다.
- 문법: const [state, setState] = useState(initialValue);

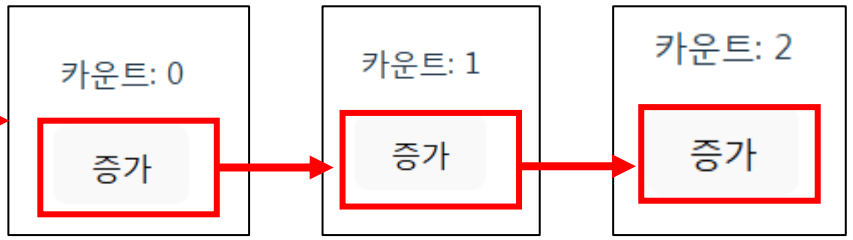


useState 간단한 예제

```jsx
import React,{useState} from 'react';


export default function Counter01() {
  const [count, setCount] = useState(0); // 초기값 0


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



기초 연습 문제

문제1] 기본 Props 전달

조건

① Welcome 컴포넌트를 완성하세요
   props로 name을 받아서 "안녕하세요, {name}님!"을 출력
② src 폴더안에 -> propsComponent 폴더 생성 -> Props01.jsx 파일을 생성하여 작성하시오.
③ src 폴더안에 -> App.jsx에 작성한 Proos02.jsx를 import하여 작성하시오

<출력 결과>

안녕하세요, 개나리님!

문제2] 여러 Props 전달

조건

① UserCard 컴포넌트를 완성하세요
   props로 name, age, city를 받아서 모두 표시
② src 폴더안에 -> propsComponent 폴더 생성 -> Props02.jsx 파일을 생성하여 작성하시오.
③ src 폴더안에 -> App.jsx에 작성한 Proos02.jsx를 import하여 작성하시오.

힌트

여러 개의 Props를 동시에 전달할 수 있습니다
숫자는 중괄호 {}로 감싸서 전달합니다 (age={25})
문자열은 따옴표로 전달합니다 (name="최영숙")

<출력 결과>

이름: 개나리
나이: 18
도시: 서울

이름: 김백합
나이: 40
도시: 경기

이름: 최영숙
나이: 25
도시: 인천

문제3] 구조 분해 할당

조건

① Product 컴포넌트를 구조 분해 할당으로 완성하세요
props.name, props.price 대신 { name, price } 형태로 받기
② src 폴더안에 -> propsComponent 폴더 생성 -> Props03.jsx 파일을 생성하여 작성하시오.
③ src 폴더안에 -> App.jsx에 작성한 Proos03.jsx를 import하여 작성하시오.

힌트

- 구조 분해 할당으로 코드가 더 간결해집니다
- props. 접두사 없이 바로 변수명으로 사용 가능
- 가독성이 향상되고 타이핑이 줄어듭니다

<출력 결과>

문제4] 기본값 설정

조건

① Button 컴포넌트에 기본값을 설정하세요
text의 기본값: "클릭", color의 기본값: "blue"

② src 폴더안에 -> propsComponent 폴더 생성 -> Props04.jsx 파일을 생성하여 작성하시오.

③ src 폴더안에 -> App.jsx에 작성한 Proos04.jsx를 import하여 작성하시오.

- 💡 **힌트**
- = 연산자를 사용해 기본값을 설정합니다
- Props가 전달되지 않으면 기본값이 사용됩니다
- Props가 전달되면 기본값은 무시됩니다

<출력 결과>

문제5] 함수 Props

조건

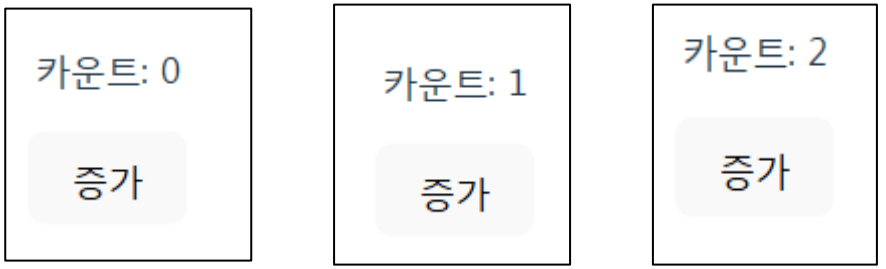
① Counter 컴포넌트를 완성하세요
count와 onIncrease를 props로 받아서 버튼에 연결

② src 폴더안에 -> propsComponent 폴더 생성 -> Props05.jsx 파일을 생성하여 작성하시오.

③ src 폴더안에 -> App.jsx에 작성한 Proos05.jsx를 import하여 작성하시오.

- 💡 **힌트**
- 함수도 Props로 전달할 수 있습니다
- 자식 컴포넌트에서 부모의 함수를 실행할 수 있습니다
- onClick={onIncrease} (호출X), onClick={onIncrease()} (호출O)

<출력 결과>



문제6] 조건부 렌더링과 Props

조건

① Alert 컴포넌트를 완성하세요
type에 따라 다른 색상 클래스 적용 (error: 빨강, success: 초록, 기타: 파랑)

② src 폴더안에 -> propsComponent 폴더 생성 -> Props06.jsx 파일을 생성하여 작성하시오.

③ src 폴더안에 -> App.jsx에 작성한 Proos06.jsx를 import하여 작성하시오.

- 💡 **힌트**
- 함수도 Props로 전달할 수 있습니다
- 자식 컴포넌트에서 부모의 함수를 실행할 수 있습니다
- onClick={onIncrease} (호출X), onClick={onIncrease()} (호출O)

<출력 결과>

문제7] 배열 Props와 map

조건

TodoList 컴포넌트를 완성하세요
todos 배열을 받아서 각 항목을 li로 표시 (key 속성 필수!)

```
const todos = [
    { id: 1, text: 'React 공부' },
    { id: 2, text: 'Props 마스터' },
    { id: 3, text: '프로젝트 만들기' },
  ];
```

① src 폴더안에 -> propsComponent 폴더 생성 -> Props07.jsx 파일을 생성하여 작성하시오.

② src 폴더안에 -> App.jsx에 작성한 Proos07.jsx를 import하여 작성하시오.

힌트

- 배열을 Props로 전달하여 동적 리스트를 생성합니다
- map() 함수로 배열의 각 항목을 JSX로 변환합니다
- **key는 필수!** 각 항목을 고유하게 식별합니다
- key는 보통 데이터의 id를 사용합니다

<출력 결과>

- React 공부
- Props 마스터
- 프로젝트 만들기