리스트 렌더링

리스트 렌더링

- 1
- 2
- 3
- 4
- 5

html 코드

```
<div>
  <ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
    <li>5</li>
  </ul>
</div>
```

- 동일한 항목이 반복하여 렌더링될 경우 리액트에서는 **리스트(배열)와 map() 함수를 이용할 수 있다.**
- **즉,** map 함수를 통해 배열의 각 요소를 반복하여 렌더링하고, key 속성을 통해 React가 각 요소를 고유하게 식별할 수 있도록 한다.

Map()

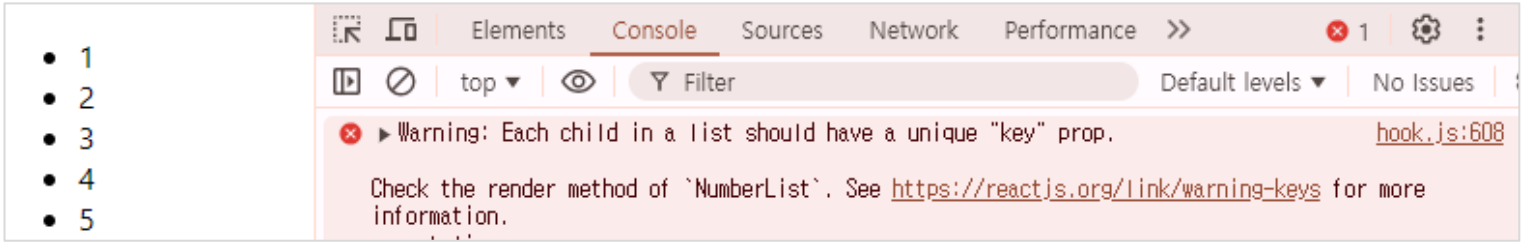
- map 함수는 배열의 각 요소를 원하는 형태로 변환하여 새로운 배열을 반환
  - map 함수를 사용하면 컴포넌트나 요소를 반복 렌더링할 때 매우 유용하다.

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) => <li>{number}</li>);
return (
  <div>
    <ul>{listItems}</ul>
  </div>
);
```

ListItem1.jsx

리스트 렌더링 주의사항

- React에서 리스트를 렌더링할 때는 각 항목에 고유한 key 속성을 부여하는 것이 좋다.
- Key 속성을 사용하지 않을 경우 경고메시지 발생




```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number, index)
                                => <li key={index}>{number}</li>);
return (
  <div>
    <ul>{listItems}</ul>
  </div>
);
```

- Key 속성을 추가하면 React가 리스트 항목을 효율적으로 관리하여 성능이 향상되고 경고 메시지가 사라짐.

Key 속성

- key 속성은 React가 각 요소를 고유하게 식별할 수 있도록 한다.
  - 고유한 key 값이 있어야만 React가 요소의 변경, 추가, 삭제를 효율적으로 처리할 수 있다.
  - key로는 각 요소마다 고유한 값을 주는 것이 좋으며, 배열의 인덱스를 사용하는 것은 권장되지 않는다. 그러나 인덱스 외에 고유한 값이 없다면 인덱스를 사용할 수 있다.
  - 예를 들어, id 값이 있는 데이터라면 다음과 같이 id 값을 key로 사용할 수 있다.

```
const items = [
  { id: 1, name: "Apple" },
  { id: 2, name: "Banana" },
  { id: 3, name: "Cherry" }
];
```

```
function ItemList() {
    const listItems = items.map((item)
        => <li key={item.id}>{item.name}</li>);
  return (
    <div>
      <ul>{listItems}</ul>
    </div>
  );
}
```

ListItem2.jsx

리스트 렌더링

- 단일 컴포넌트(함수) 내에서 리스트 렌더링 하는 경우 바로 렌더링하므로 코드가 간결할 수 있으나 유연성은 떨어진다.

  - **코드 재사용 어려움.(배열 값이 변경되려면 직접 수정해야 한다.)**

- props로 데이터를 전달하여 재사용 할 수 있도록 한다.
- ListItems 컴포넌트를 사용하면 다른 부분에서 이 컴포넌트를 재사용할 수 있다.(**재사용성, 유연성**)

리스트 렌더링

- ListItems라는 별도의 컴포넌트를 정의하고, 배열을 props로 전달 받는다.
- ListItems 컴포넌트를 사용하면 다른 부분에서 이 컴포넌트를 재사용할 수 있다.(**재사용성, 유연성**)

ListItems.jsx

```
function ListItems(props) {
  const { items } = props;

  const listItems = items.map((item) =>
              <li key={item.id}>{item.name}</li>);
  return (
    <div>
      <ul>{listItems}</ul>
    </div>
  );
}
```

리스트 렌더링

- ListIRender 컴포넌트에서 ListItems 컴포넌트 호출하면서 리스트 데이터를 전달

```
const numbers = [
  { id: 1, name: "one" },
  { id: 2, name: "two" },
  { id: 3, name: "three" },
  { id: 4, name: "four" },
  { id: 5, name: "five" },
];
const fruits = [
  { id: 1, name: "apple" },
  { id: 2, name: "banana" },
  { id: 3, name: "kiwi" },
  { id: 4, name: "mango" },
  { id: 5, name: "pineapple" },
];
```

```
function ListRender() {
  return (
    <div>
      <ListItems items={numbers} />
      <ListItems items={fruits} />
    </div>
  );
}
```

ListRender.jsx

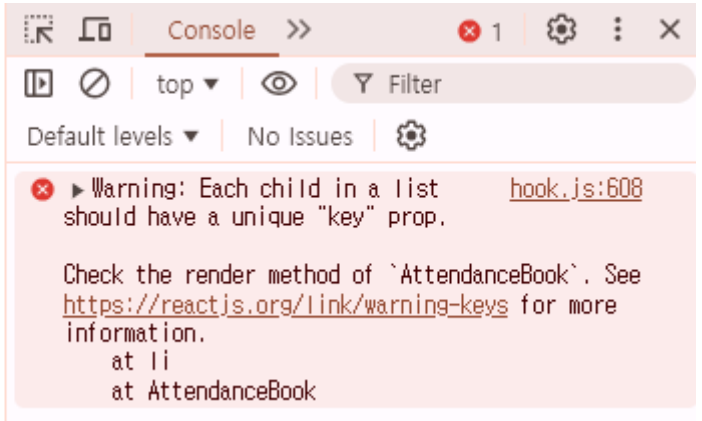
실습: 출석부 출력

map() 함수 안에 사용되는 데이터 항목은 딕셔너리 구조로 작성해야 한다

```
const students = [
  {  name: "Inje", },
  {  name: "Steve",},
  {  name: "Bill", },
  {  name: "Jeff", },
];
```

```
function AttendanceBook(props) {
  return (
    <ul>
      {students.map((student) => {
        return <li>{student.name}</li>;
      })}
    </ul>
  );
}
```

- Inje
- Steve
- Bill
- Jeff



map() 함수 안에 있는 엘리먼트는 꼭 키가 있어야 한다. 아닐 경우 warning 발생

실습: 출석부 출력

- Inje
- Steve
- Bill
- Jeff

```
const students = [
  {
    id: 1,
    name: "Inje",
  },
  {
    id: 2,
    name: "Steve",
  },
  {
    id: 3,
    name: "Bill",
  },
  {
    id: 3,
    name: "Jeff",
  },
];
```

```
function AttendanceBook(props) {
  return (
    <ul>
      {students.map((student, index) => {
        return <li key={student.id}>{student.name}</li>;
      })}
    </ul>
  );
}
```

실습: 출석부 출력

Map() 함수안에서 키 사용 예시

```
01  // id를 키값으로 사용
02  {students.map((student) => {
03      <li key={student.id}>{student.name}</li>;
04  })}
05
06  // 포맷팅 된 문자열을 키값으로 사용
07  {students.map((student, index) => {
08      <li key={`student-id-${student.id}`}>{student.name}</li>;
09  })}
10
11  // 배열의 인덱스를 키값으로 사용
12  {students.map((student, index) => {
13      <li key={index}>{student.name}</li>;
14  })}
```

실습1 : 학생 명단을 출력하는 코드 작성(리스트 렌더링)

1.1 StudentList.컴포넌트로 만들기

```
const students = [
        { id: 1, name: "Alice" },
        { id: 2, name: "Bob" },
        { id: 3, name: "Charlie" },
  ];

function StudentList() {
  이곳에 알맞은 코드 작성
  return (
    <div>
      <h2>학생 명단</h2>
      <ul>{ list }</ul>
    </div>
  );
}
```

1.2 ListRender 컴포넌트에서 <StudentList items={studens}/> 호출할 수 있도록 코드 수정하기

실습2 : 학생 정보 출력하는 코드 작성(리스트 렌더링)

학생 정보

번호	이름	이메일	성별
1	Alice	alice@example.com	Female
2	Bob	bob@example.com	Male
3	Charlie	charlie@example.com	Male

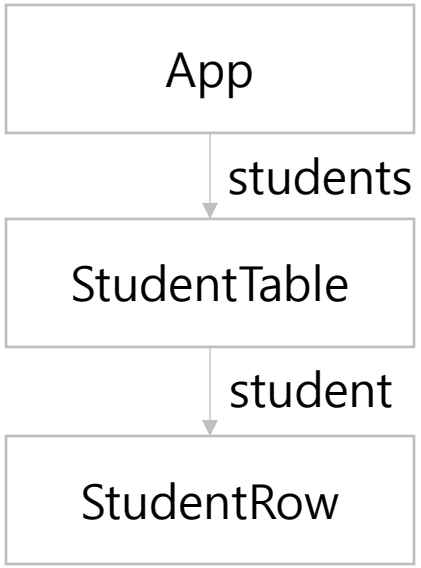
```
Function StudentTable () {
  const students = [
    { id: 1, name: "Alice", email: "alice@example.com", gender: "Female" },
    { id: 2, name: "Bob", email: "bob@example.com", gender: "Male" },
    { id: 3, name: "Charlie", email: "charlie@example.com", gender: "Male" },
  ];

  return (
    <div>
      <h2>학생 정보</h2>
      <table border="1">
        <thead>
          <tr>
            <th>번호</th><th>이름</th><th>이메일</th><th>성별</th>
          </tr>
        </thead>
        <tbody>
            { studentRow }
        </tbody>
      </table>
    </div>
  );
}
```

실습2: 학생 정보 리스트 렌더링

```
Const studentRow =
          students.map((student) => (
            <tr key={student.id}>
              <td>{student.id}</td>
              <td>{student.name}</td>
              <td>{student.email}</td>
              <td>{student.gender}</td>
            </tr>
          ))
```

실습3 : 학생 정보 리스트 렌더링 코드 분해 및 컴포넌트화

- 테이블 전체를 StudentInfo 컴포넌트로 만들기
- 각 학생의 정보를 표시하는 StudentRow 컴포넌트를 별도로 분리하기



App: 학생 정보를 포함한 students 데이터를 정의하고 이를 StudentTable 컴포넌트에 전달한다.

StudentTable: App으로부터 students 배열을 props로 받아 테이블 구성. 각 학생 데이터에 대해 StudentRow 컴포넌트를 호출하여 한 행씩 렌더링한다.

StudentRow: StudentTable에서 전달받은 각 학생의 데이터를 이용해 <tr> 요소로 정보를 렌더링한다.

```
const studentRow = students.map((student) => (
  <StudentRow key={student.id} student={student} />
));
```

실습3 : 학생 정보 리스트 렌더링 코드 분해 및 컴포넌트화

```
function App() {
  const students = [
    { id: 1, name: "Alice", email: "alice@example.com", gender: "Female" },
    { id: 2, name: "Bob", email: "bob@example.com", gender: "Male" },
    { id: 3, name: "Charlie", email: "charlie@example.com", gender: "Male" },
  ];

  return (
    <div>
      <StudentTable students={students} />
    </div>
  );
}
```

실습3: 학생 정보 리스트 렌더링 코드 분해 및 컴포넌트화

StudentTable 컴포넌트

```
<tbody>
  {students.map((student) => (
    <tr key={student.id}>
      <td>{student.id}</td>
      <td>{student.name}</td>
      <td>{student.email}</td>
      <td>{student.gender}</td>
    </tr>
  ))}
</tbody>
```

StudentRow 컴포넌트

```
StudentRow(props){

  const { student } = props;

  return (   )
}
```

```
<tbody>
  {students.map((student) => (
  <StudentRow key={student.id} student={student} />
  ))}
</tbody>
```

StudentRow 컴포넌트는 각 학생의 정보를
하나의 행으로 렌더링된다.

CSS 적용

```
import "./StudentTable.css";
```

학생 정보

번호	이름	이메일	성별
1	Alice	alice@example.com	Female
2	Bob	bob@example.com	Male
3	Charlie	charlie@example.com	Male

```css
.student-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
}

.student-table th,
.student-table td {
  border: 1px solid #ddd;
  padding: 8px;
  text-align: left;
}

.student-table th {
  background-color: #f2f2f2;
  color: #333;
  font-weight: bold;
}

.student-table tr:nth-child(even) {
  background-color: #f9f9f9;
}

.student-table tr:hover {
  background-color: #e2e2e2;
}
```

Form 입력양식

Form

- 사용자 입력 양식

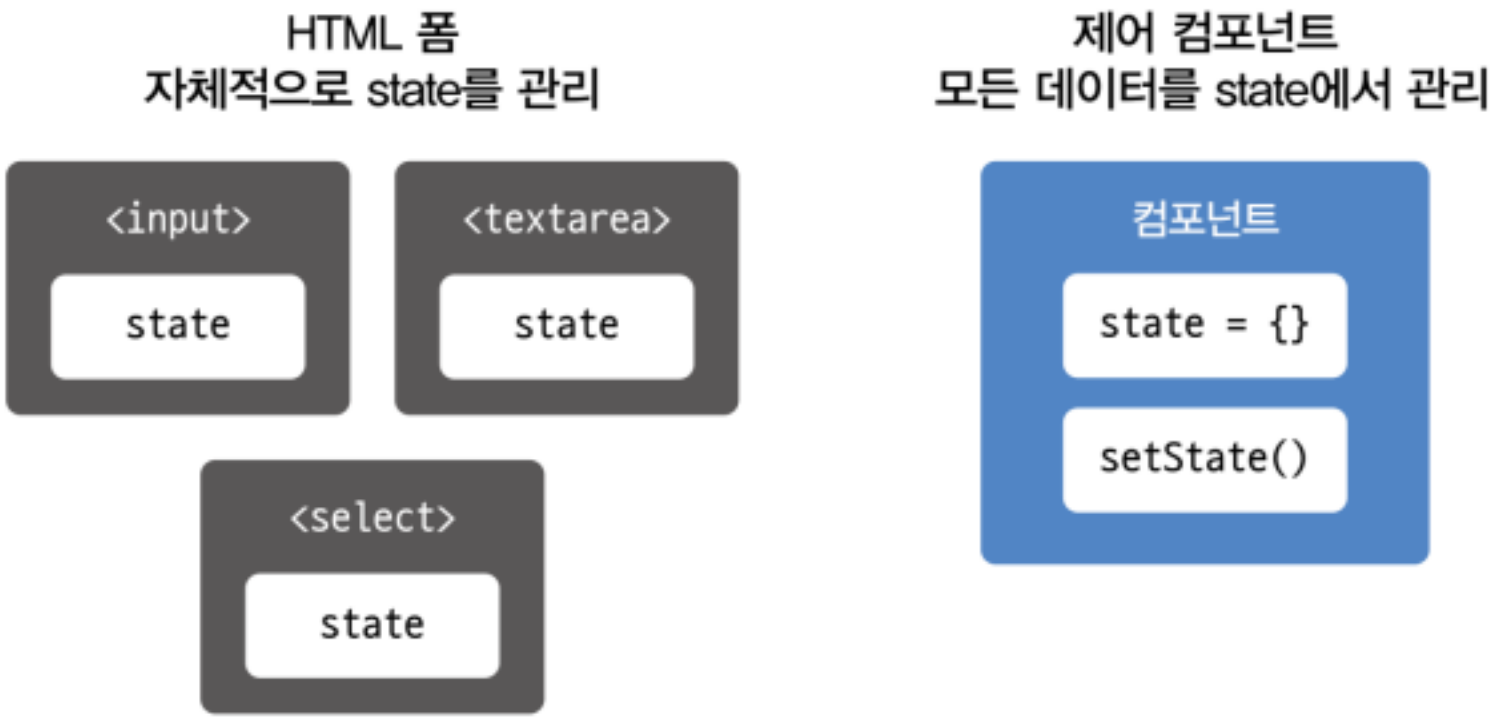
[HTML 폼 양식 예]

```
01  <form>
02      <label>
03          이름:
04          <input type="text" name="name" />
05      </label>
06      <button type="submit">제출</button>
07  </form>
```

- HTML 폼은 각 엘리먼트가 자체적으로 state 관리하므로 자체적으로 내부에 state를 가짐
- 자바스크립트 코드로 각 엘리먼트 값 접근하기 불편

사용자 입력값 접근 제어하기

- 제어 컴포넌트(controlled component)
  - 리액트에 의해 통제되는 폼 엘리먼트



- HTML 폼은 각 엘리먼트가 **자체적으로 state 관리**하므로 자체적으로 **내부 state를 가짐**
- 자바스크립트 코드로 각 엘리먼트 값 접근하기 불편
- 제어 컴포넌트에서는 useState() 이용하여 모든 데이터 관리

사용자 입력값 접근 제어하기

- 제어 컴포넌트(controlled component)
  - input 예시

이름: 제출

```
01  function NameForm(props) {
02      const [value, setValue] = useState('');
03
04      const handleChange = (event) => {
05          setValue(event.target.value);
06      }
07
08      const handleSubmit = (event) => {
09          alert('입력한 이름: ' + value);
10          event.preventDefault();
11      }
12
13      return (
14          <form onSubmit={handleSubmit}>
15              <label>
16                  이름:
17                  <input type="text" value={value} onChange={handleChange} />
18              </label>
19              <button type="submit">제출</button>
20          </form>
21      )
22  }
```

사용자 입력값 접근 제어하기

- 제어 컴포넌트(controlled component) 예
  - 사용자 입력한 모든 알파벳을 대문자로 변환

```
01   const handleChange = (event) => {
02       setValue(event.target.value.toUpperCase());
03   }
```

사용자 입력값 접근 제어하기

- <textarea></textarea>

```
<textarea>
    안녕하세요, 여기에 이렇게 텍스트가 들어가게 됩니다.
</textarea>
```

```
01  function RequestForm(props) {
02      const [value, setValue] = useState('요청사항을 입력하세요.');
03
04      const handleChange = (event) => {
05          setValue(event.target.value);
06      }
07
08      const handleSubmit = (event) => {
09          alert('입력한 요청사항: ' + value);
10          event.preventDefault();
11      }
12
13      return (
```

사용자 입력값 접근 제어하기

- <textarea></textarea>

```
14          <form onSubmit={handleSubmit}>
15              <label>
16                  요청사항:
17                  <textarea value={value} onChange={handleChange} />
18              </label>
19              <button type="submit">제출</button>
20          </form>
21      )
22  }
```

Select 태그

- 드롭다운 목록을 보여주기 위한 html 태그
  - <option>의 selected 성에 현재 선택된 값 지정됨

```
<select>
    <option value="apple">사과</option>
    <option value="banana">바나나</option>
    <option selected value="grape">포도</option>
    <option value="watermelon">수박</option>
</select>
```

Select 태그

- 리액트에서는 <select> 태그에 value라는 attribute 사용하여 값을 표시하고 useSttate로 상태값을 관리한다.

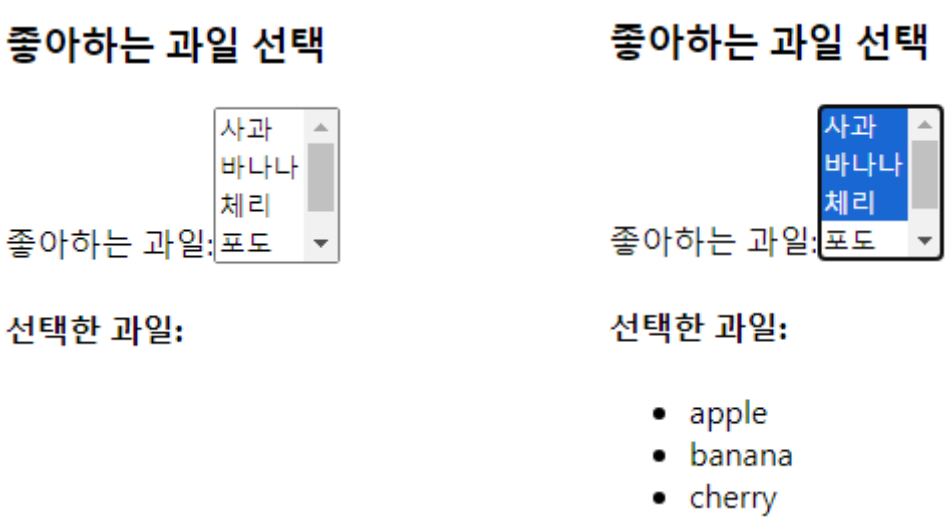
```
function FruitSelect(props) {
    const [value, setValue] = useState('grape');

    const handleChange = (event) => {
        setValue(event.target.value);
    }

const handleSubmit = (event) => {
    alert('선택한 과일: ' + value);
    event.preventDefault();
}
```

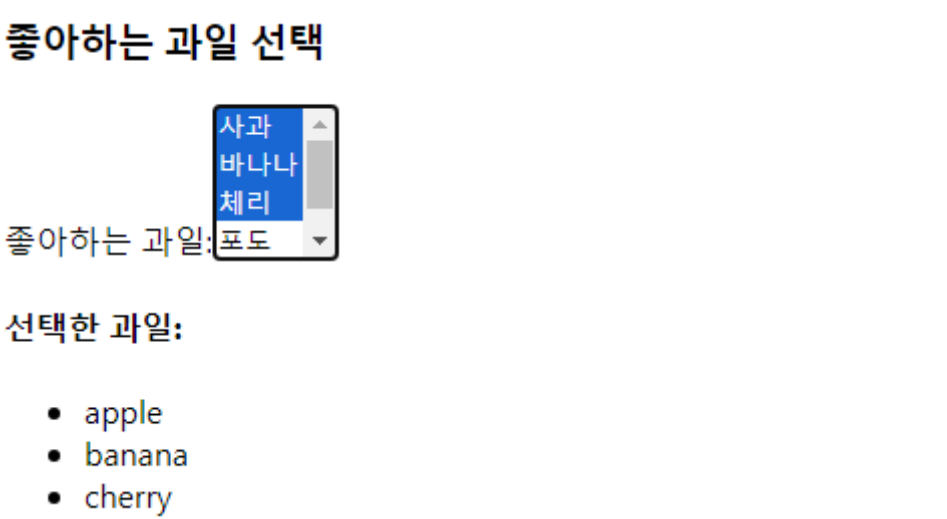
```
<form onSubmit={handleSubmit}>
    <label>
        과일을 선택하세요:
        <select value={value} onChange={handleChange}>
            <option value="apple">사과</option>
            <option value="banana">바나나</option>
            <option value="grape">포도</option>
            <option value="watermelon">수박</option>
        </select>
    </label>
    <button type="submit">제출</button>
</form>
```

Select 태그-다중선택

- 다중 선택이 되도록 하기 위해서는 multiple 속성을 true로 하고, value 로 선택된 옵션의 값이 들어있는 배열을 넣어준다.

```
<select multiple={true} value={['B', 'C']}>
```





Select 태그-다중선택

- 다중 선택이 되도록 하기 위해서는 multiple 속성을 true로 하고, value 로 선택된 옵션의 값이 들어있는 배열을 넣어준다.

```
function FavoriteFruitsForm() {
  const [selectedFruits, setSelectedFruits] = useState([]);

  const handleChange = (event) => {
    const selectedOptions = [...event.target.selectedOptions].map(
      (option) => option.value
    );
    setSelectedFruits(selectedOptions);
  };
```

Select 태그

```
  return (
    <div>
      <h3>좋아하는 과일 선택</h3>
      <form>
        <label>
          좋아하는 과일:
          <select multiple={true} value={selectedFruits} onChange={handleChange} >
            <option value="apple">사과</option>
            <option value="banana">바나나</option>
            <option value="cherry">체리</option>
            <option value="grape">포도</option>
            <option value="orange">오렌지</option>
          </select>
        </label>
      </form>
```

```
        <div>
          <h4>선택한 과일:</h4>
          <ul>
            {selectedFruits.map((fruit) => (
              <li key={fruit}>{fruit}</li>
            ))}
          </ul>
        </div>
      </div>
    );
}
```

복수개 입력 다루기

```
01  function Reservation(props) {
02      const [haveBreakfast, setHaveBreakfast] = useState(true);
03      const [numberOfGuest, setNumberOfGuest] = useState(2);
04
05      const handleSubmit = (event) => {
06          alert(`아침식사 여부: ${haveBreakfast}, 방문객 수: ${numberOfGuest}`);
07          event.preventDefault();
08      }
09
10      return (
11          <form onSubmit={handleSubmit}>
12              <label>
13                  아침식사 여부:
14                  <input
15                      type="checkbox"
16                      checked={haveBreakfast}
17                      onChange={(event) => {
18                          setHaveBreakfast(event.target.checked);
19                      }} />
20              </label>
21              <br />
22              <label>
23                  방문객 수:
24                  <input
25                      type="number"
26                      value={numberOfGuest}
27                      onChange={(event) => {
28                          setNumberOfGuest(event.target.value);
29                      }} />
30              </label>
31              <button type="submit">제출</button>
32          </form>
33      );
34  }
```

실습(교재): 사용자 입력 받기

실습(응용)

사용자 정보 입력

이름:
성별: ○ 남성 ○ 여성
나이:
좋아하는 과일: 선택하세요
제출

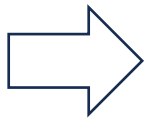
입력한 사용자 정보

이름:

성별:

나이:

좋아하는 과일:



사용자 정보 입력

이름: 홍길동
성별: ◉ 남성 ○ 여성
나이: 24
좋아하는 과일: 체리
제출

입력한 사용자 정보

이름: 홍길동

성별: 남성

나이: 24

좋아하는 과일: 체리