Seoul Software ACademy

웹 개발자 부트캠프 과정

SeSAC x CODINGOn

With. 팀 뤼쳐드

map()을 이용한 반복

map() 함수 문법

```
arr.map( callbackFunction, [thisArg] )
```

- callbackFunction
  - 새로운 배열의 요소를 생성하는 함수로, currentValue, index, array 3개의 인수를 가질 수 있다.
- [this.Arg] 는 생략 가능한 것으로 callbackFunction 에서 사용할 this 객체

배열 데이터, map()과 함께 사용

- 배열 데이터를 좀 더 효율적으로 그리기 위해서 map 사용!

```
{배열.map((요소, 인덱스) => {
        return <div key={인덱스}>{요소}</div>;
})}
```

- key
  - 기존 요소와 업데이트 요소를 비교하는데 사용되는 속성,
  - 다른요소와 겹치지 않는 고유한 값이어야 합니다.
  - key는 고유한 값을 가져야 하기 때문에 배열의 요소 중 고유한 값(id 등)이 존재하지 않는다면 index로 사용해도 됩니다! (단, index 를 사용하는 것은 최후의 수단!)

배열 데이터, map()과 함께 사용

- div 같은 태그가 아닌 만들어준 컴포넌트와 함께 사용할 수도 있겠죠!

```
{배열.map((element, index) => {
    return <FunctionProps name={element.name} key={index} />;
  })}
```

let 배열=[{name:"allie"},
{name:"Lucy"},
{name:"Linda"},
];
와 같은 오브젝트로 이루어진 배열이라면

오브젝트의 점 접근법으로 props 전달 &
배열의 개수만큼 컴포넌트 생성,

Component에 map() 적용

```
const [list, setList] = useState(['a','b','c','d','e']);

return (
    <>
    <ol>
        {list.map((value)=>{
            return <li>{value}</li>
        })}
    </ol>
    </>
);
```

```
<ol>
    {list.map((value) => <li>{value}</li>)}
</ol>
```

1. a
2. b
3. c
4. d
5. e

Component에 map() 적용

```
▶Warning: Each child in a list should have a unique "key" prop.    react-refresh-runtime.development.js:688

Check the render method of `MapTest`. See https://reactjs.org/link/warning-keys for more information.
    at li
    at MapTest (http://localhost:3000/main.f503859….hot-update.js:30:74)
    at App
```

- map() 함수를 이용해 컴포넌트를 생성할 때 “key” 사용을 권장한다.
- Why? React는 업데이트 전 기존 요소와 업데이트 후 요소를 비교하는데 key를 사용

Component에 map() 적용

```
<li key={id}>{value}</li>
```

- Key를 index 값으로 설정할 시, 리스트의 순서가 변경되면 모든 key가 변경되므로 key는 index 가 아닌 고유한 값으로 설정해야 한다
  - But, 현재는 **고유 값으로 설정할 만한 게 없으니, index**로 테스트

Component에 map() 적용

- 각 원소마다 고유 id 값을 갖고 있다면? 다음과 같이 설정할 수 있다!

```
const [list, setList] = useState([
    { id: 1, alpha: 'a'},
    { id: 2, alpha: 'b'},
    { id: 3, alpha: 'c'},
    { id: 4, alpha: 'd'},
    { id: 5, alpha: 'e'},
]);

return (
    <>
    <ol>
        {list.map((value) => <li key={value.id}>{value.alpha}</li>)}
    </ol>
    </>
);
```

Component에 map() 적용

- input으로 새로운 알파벳 추가해보기

Component에 map() 적용

- input으로 새로운 알파벳 추가해보기

```
const [inputAlpha, setInputAlpha] = useState('');
const addAlpha = () => {
    // concat() 메서드는 인자로 주어진 값을 기존 배열에 합쳐서 새 배열을 반환.
    const newAlpha = list.concat({id: list.length+1, alpha: inputAlpha});
    setList(newAlpha);
    setInputAlpha('');
}

return (
    <>
    <input value={inputAlpha} onChange={(e)=>{setInputAlpha(e.target.value);}}></input>
    <button onClick={addAlpha}>추가</button>
    <ol>
        {list.map((value) => <li key={value.id}>{value.alpha}</li>)}
    </ol>
    </>
);
```

filter()를 이용한 필터링

filter() 함수

- filter()의 인자로 넘겨지는 callback 함수의 테스트(조건)를 통과하는 요소를 모아 새로운 배열을 생성.
- filter() 함수를 사용하여 배열에서 원하는 값을 삭제하는 코드 구현 가능.

filter() 함수

```
let animals = ['dog', 'cat', 'rabbit'];

let newAnimals = animals.filter((animal)=>{ return animal.length > 3});
console.log(newAnimals);
```


```
let newAnimals = animals.filter((animal)=> animal.length > 3);
```

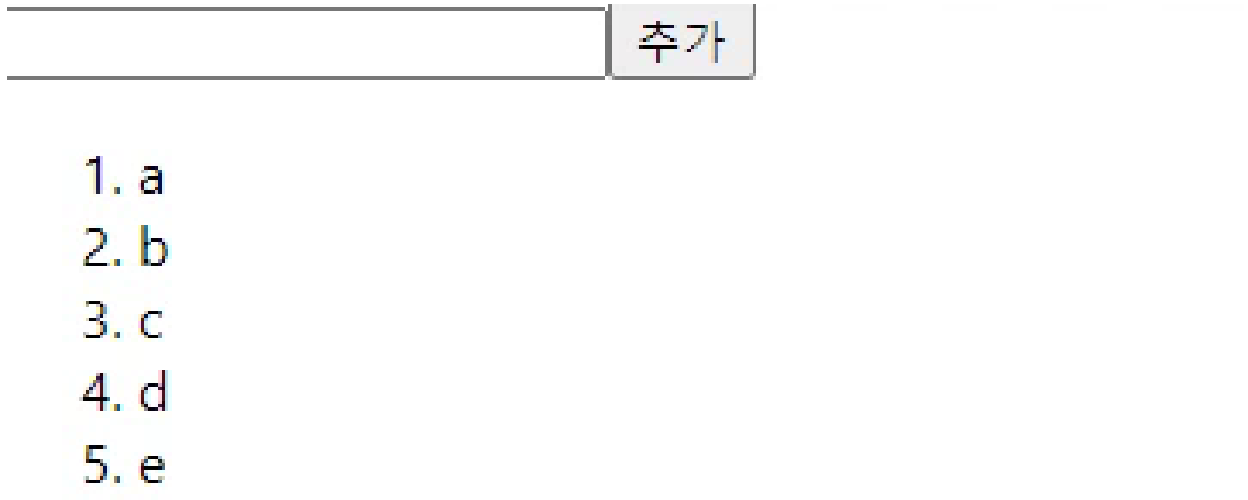
filter() 함수

```
let words = ['dog', 'cat', 'rabbit'];

let result2 = words.filter((word) => {
    return word.includes('a');
});
console.log( result2 );
```

filter() 응용

- 알파벳을 더블 클릭 했을 때 알파벳 삭제해보기



filter() 응용

```
const [list, setList] = useState([ ···
]);
const [inputAlpha, setInputAlpha] = useState('');
const addAlpha = () => {···
}

const deleteAlpha = (id) => {
    const newAlpha = list.filter((value)=> value.id !== id);
    setList(newAlpha);
}

return(
    <>
    <input value={inputAlpha} onChange={(e)=>{setInputAlpha(e.target.value);}}></input>
    <button onClick={addAlpha}>추가</button>
    <ol>
        {list.map((value) => <li key={value.id} onDoubleClick={()=>{deleteAlpha(value.id)}} >{value.alpha}</li>)}
    </ol>
    </>
);
```