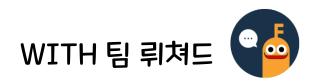


K-Digital Training KDT 풀스택 웹 개발자 양성 부트캠프 3기

React









일반 for 반복문

```
let list = ['a','b','c','d','e'];
for ( let i = 0; i < list.length; i++ ) {
    console.log( list[i] );
}</pre>
```



map() 함수

- map()의 인자로 넘겨지는 callback 함수를 실행한 결과를 가지고 새로 운 배열을 만들 때 사용.
- map() 함수를 필요에 따라 반복문처럼 사용할 수도 있음.

코드와 함께 알아보자!



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



map() 함수

```
let list = ['a','b','c','d','e'];
let items = list.map((txt, id, arr)=>{
    console.log("txt: ", txt);
    console.log("id: ", id);
    console.log("arr: ", arr);
    return txt + id;
})
```



map() 함수

```
let list = ['a','b','c','d','e'];
let items = list.map((txt, id, arr)=>{
    console.log("txt: ", txt);
    console.log("id: ", id);
    console.log("arr: ", arr);
    return txt + id;
})
```

- txt : list 를 순서대로 돌면서 나오는 값
- id : 방금 나온 값(txt)의 인덱스
- arr : 현재 반복을 돌고 있는 배열
- items: "return txt + id;" 로 만들어진 배열



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

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

1. a

2. b

3. c

4. d

5. e



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

Check the render method of `MapTest`. See <a href="https://reactjs.org/link/warning-keys">https://reactjs.org/link/warning-keys</a> for more information.

at li
at MapTest (<a href="http://localhost:3000/main.f503859....hot-update.js:30:74">http://localhost:3000/main.f503859....hot-update.js:30:74</a>)
at App
```

- map() 함수를 이용해 컴포넌트를 생성할 때 "key" 사용을 권장한다.
- Why? React는 자율적으로 업데이트 전 기존 요소와 업데이트 요소를 비교하는데 key를 사용한다.



```
key={id}>{value}
```

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



• 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 (
   <>
   {list.map((value) => {value.alpha})}
   </>>
```



• input으로 새로운 알파벳 추가해보기

추가

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



• 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>
   {list.map((value) => {value.alpha}}}
   </>>
```



filter()



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.

코드와 함께 알아보자!



filter() 함수

```
let animals = ['dog', 'cat', 'rabbit'];
let newAnimals = animals.filter((animal)=>{ return animal.length > 3});
console.log(newAnimals);
```

```
▶ ['rabbit']
```

```
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

추가

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



filter() 응용

```
const [list, setList] = useState([ ...
1);
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>
   {list.map((value) => {deleteAlpha(value.id)}} >{value.alpha}
   </>>
```