

WebShooter

WebShooter는 웹 개발을 위한 문제 생성 및 채점 시스템으로
실시간으로 문제를 풀고 피드백을 받을 수 있는 플랫폼입니다.
다양한 웹 개발 이론과 기술을 학습하고, 문제를 해결해보세요!

[문제 생성하기](#)[전체 실습 문제 >](#)

Level 1

[Button UI 생성하기](#)

JSX

Level 3

[갤러리 에러 수정](#)

state

Event Handling

Level 4

[Form Validation](#)

zod

schema

Level 2

[컴포넌트 생성](#)

JSX

Level 5

[채팅방 재연결 로직](#)

커스텀 흑

Effect

Level 3

[갤러리 에러 수정](#)

state

Event Handling

WebShooter

WebShooter는 웹 개발을 위한 문제 생성 및 채점 시스템으로
실시간으로 문제를 풀고 피드백을 받을 수 있는 플랫폼입니다.
다양한 웹 개발 이론과 기술을 학습하고, 문제를 해결해보세요!

[문제 생성하기](#)[전체 실습 문제 >](#)

Level 1

[Button UI 생성하기](#)

JSX

Level 3

[갤러리 에러 수정](#)

state

Event Handling

Level 4

[Form Validation](#)

zod

schema

Level 2

[컴포넌트 생성](#)

JSX

Level 5

[채팅방 재연결 로직](#)

커스텀 흑

Effect

Level 3

[갤러리 에러 수정](#)

state

Event Handling

WebShooter

X

문제 생성하기

기술 유형



직접 입력

추가

난이도

- Level 1 Level 2 Level 3
 Level 4 Level 5

생성

전체 실습 문제 >

Level 1

Button UI 생성하기

JSX

Level 2

컴포넌트 생성

JSX

Level 4

Form Validation

schema

Level 5

채팅방 재연결 로직

커스텀 헥 Effect

Level 3

갤러리 에러 수정

state Event Handling

점 시스템으로
풀랫폼입니다.
해결해보세요!

WebShooter

X

문제 생성하기

기술 유형

- JSX
- State
- Event Handling
- Effect
- UI
- UX
- schema
- Controller
- Domain
- Repository
- Unit Test
- HTTP
- 커스텀 헥
- Form
- Service
- zod

추가Prop XComponent X

난이도

- Level 1
- Level 2
- Level 3
- Level 4
- Level 5

생성

점 시스템으로
풀랫폼입니다.
해결해보세요!

[전체 실습 문제 >](#)

Level 1

Button UI 생성하기

JSX

Level 2

컴포넌트 생성

JSX

Level 5

채팅방 재연결 로직

커스텀 헥 Effect

Level 3

갤러리 에러 수정

state Event Handling

WebShooter

WebShooter

실시간으로 문

다양한 웹 개발

전체 실습 문제 >

Level 1

Button UI 생성하기

JSX

비밀번호 찾기

회원가입

14

Form Validation

zod schema

Level 2

컴포넌트 생성

JSX

Level 5

채팅방 재연결 로직

커스텀 흑 Effect

Level 3

갤러리 에러 수정

state Event Handling



WebShooter

점 시스템으로
플랫폼입니다.
해결해보세요!



이메일

비밀번호

로그인

회원가입

14

Form Validation

zod schema

Level 3

갤러리 에러 수정

state Event Handling

회원가입

X

아이디
아이디를 입력해 주세요

이메일
이메일을 입력해 주세요

비밀번호
영문자, 숫자, 특수문자 포함 8-20자

비밀번호를 확인해주세요

관심 분야
풀스택 프론트엔드 백엔드

관심 언어
Java JS TS Python Ruby PHP HTML CSS

회원가입

WebShooter
실시간으로
다양한 웹 개발

전체 실습 문제 >

Level 1

Button UI 생성하기

JSX

회원가입

Level 2

컴포넌트 생성

JSX

Level 5

채팅방 재연결 로직

커스텀 흑 Effect

Level 3

갤러리 에러 수정

state Event Handling

점 시스템으로
플랫폼입니다.
해결해보세요!

14

Form Validation

schema

WebShooter

WebShooter는 개발자에게 필요한 다양한 웹 기술을
실시간으로 둘러보면서 배울 수 있는
다양한 웹 개발 실습 문제입니다.

비밀번호 재설정

비밀번호 재설정 시스템으로
플랫폼입니다.
해결해보세요!

이메일 입력

이메일을 입력해 주세요

인증하기

인증코드

인증코드를 입력해 주세요

확인

전체 실습 문제 >

Level 1

Button UI 생성하기

JSX

Level 2

갤러리 에러 수정

state Event Handling

Level 3

Form Validation

zod schema

Level 4

컴포넌트 생성

JSX

Level 5

채팅방 재연결 로직

커스텀 흐 Effect

Level 6

갤러리 에러 수정

state Event Handling

WebShooter

WebShooter는 개발자에게 필요한 다양한 웹 기술을
실시간으로 둘러보면서 미션 시스템으로
다양한 웹 개발 문제를 해결해보세요!

전체 실습 문제 >

Level 1

Button UI 생성하기

JSX

수정

Level 2

갤러리 에러 수정

state

Event Handling

Level 3

Form Validation

zod

schema

Level 4

컴포넌트 생성

JSX

Level 5

채팅방 재연결 로직

커스텀 흐름

Effect

비밀번호 재설정

새 비밀번호

새 비밀번호를 입력해 주세요

새 비밀번호 확인

새 비밀번호를 다시 입력해 주세요

X

돌랫폼입니다.

해결해보세요!

전체 32건

기술 유형

난이도 ▾

Level 1

Button UI 생성하기

JSX

Level 2

컴포넌트 생성

JSX

Level 3

갤러리 에러 수정

state

Event Handling

Level 4

Form Validation

zod

schema

Level 5

채팅방 재연결 로직

커스텀 흐

Effect

Level 3

재사용 가능한 컴포넌트

JSX

Level 3

고급 이벤트 처리

JSX

Level 3

다크 모드 및 동적 스타일링

UI

UX

Level 4

상태 관리와 버튼 연동

JSX

Level 4

Custom Hooks

JSX

Level 5

디자인 시스템 컴포넌트 통합

JSX

Level 5

애니메이션과 접근성 향상

JSX



전체 32건

기술 유형

난이도 ▾

기술 유형

JSX State Event Handling 커스텀 훅

Effect UI UX zod schema

Prop Component Controller Service

Domain Repository Unit Test Form

HTTP

세팅방 세션설정 도식

커스텀 훅 Effect

세션용 기능안 검포넌트

JSX

Level 3

갤러리 에러 수정

state Event Handling

Level 4

Form Validation

zod schema

Level 3

고급 이벤트 처리

JSX

Level 3

다크 모드 및 동적 스타일링

UI UX

Level 4

상태 관리와 버튼 연동

JSX

Level 4

Custom Hooks

JSX

Level 5

디자인 시스템 컴포넌트 통합

JSX

Level 5

애니메이션과 접근성 향상

JSX



전체 32건

기술 유형

난이도 ▾

Level 1

Button UI 생성하기

JSX

Level 2

생성

- Level 1
- Level 2
- Level 3
- Level 4
- Level 5

JSX

Level 3

갤러리 에러 수정

state

Event Handling

Level 4

Form Validation

zod

schema

Level 5

채팅방 재연결 로직

커스텀 흐

Effect

Level 3

재사용 가능한 컴포넌트

JSX

Level 3

고급 이벤트 처리

JSX

Level 3

다크 모드 및 동적 스타일링

UI

UX

Level 4

상태 관리와 버튼 연동

JSX

Level 4

Custom Hooks

JSX

Level 5

디자인 시스템 컴포넌트 통합

JSX

Level 5

애니메이션과 접근성 향상

JSX



전체 8건

JSX

난이도 ▾

Level 1

Button UI 생성하기

JSX

Level 2

컴포넌트 생성

JSX

Level 3

재사용 가능한 컴포넌트

JSX

Level 3

고급 이벤트 처리

JSX

Level 4

상태 관리와 버튼 연동

JSX

Level 4

Custom Hooks

JSX

Level 5

디자인 시스템 컴포넌트 통합

JSX

Level 5

애니메이션과 접근성 향상

JSX



전체 6건

[기술 유형](#)[Level 3 ▾](#)

Level 3

갤러리 에러 수정[state](#) [Event Handling](#)

Level 3

재사용 가능한 컴포넌트[JSX](#)

Level 3

고급 이벤트 처리[JSX](#)

Level 3

라우팅 처리[UX](#)

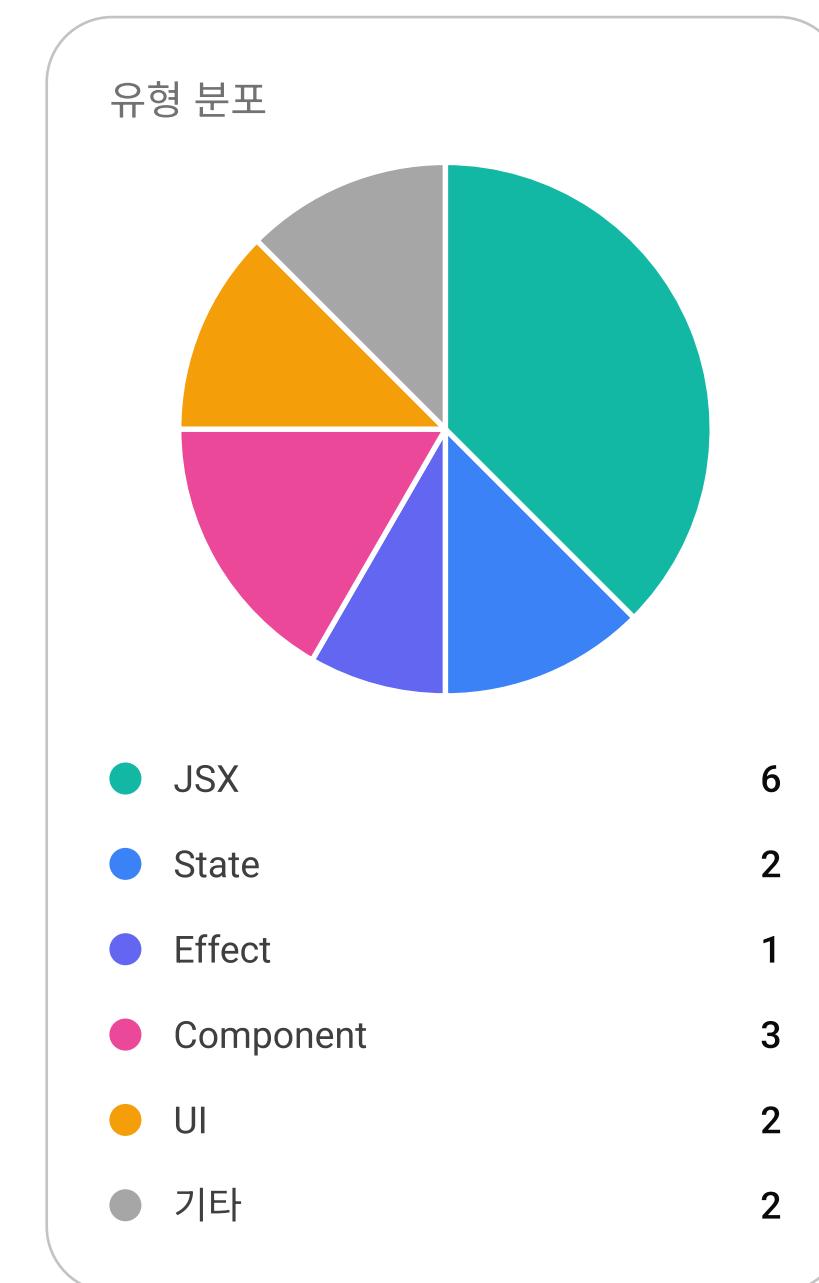
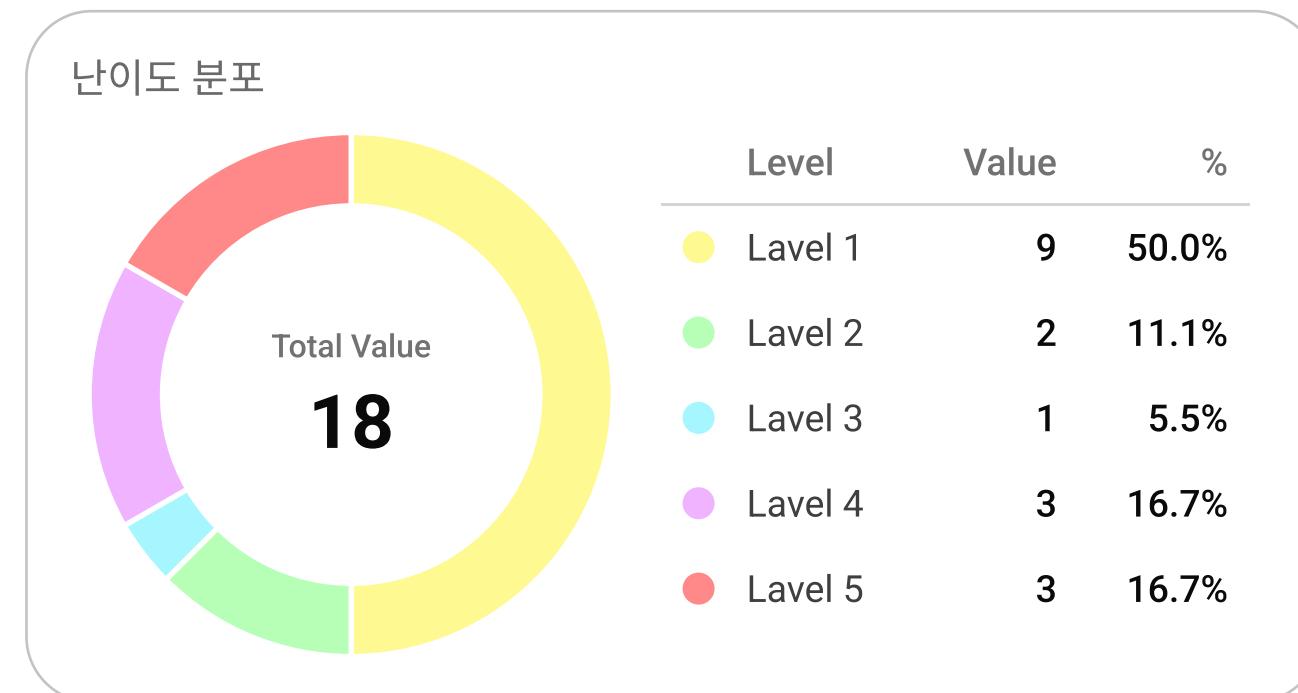
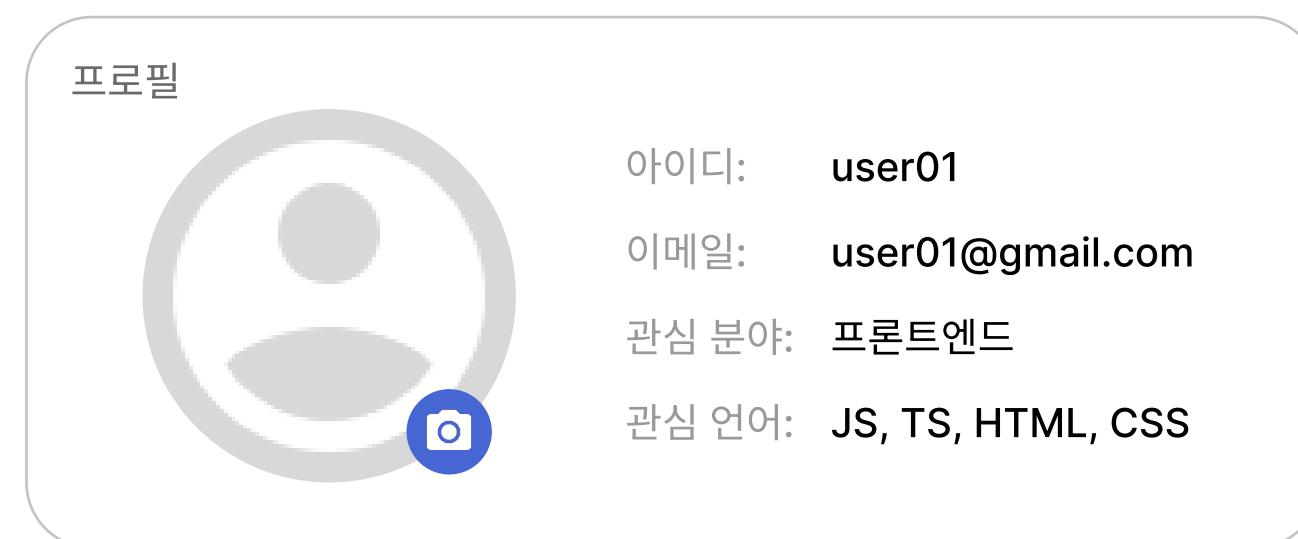
Level 3

전역 상태 관리[Redux](#) [Zustand](#)

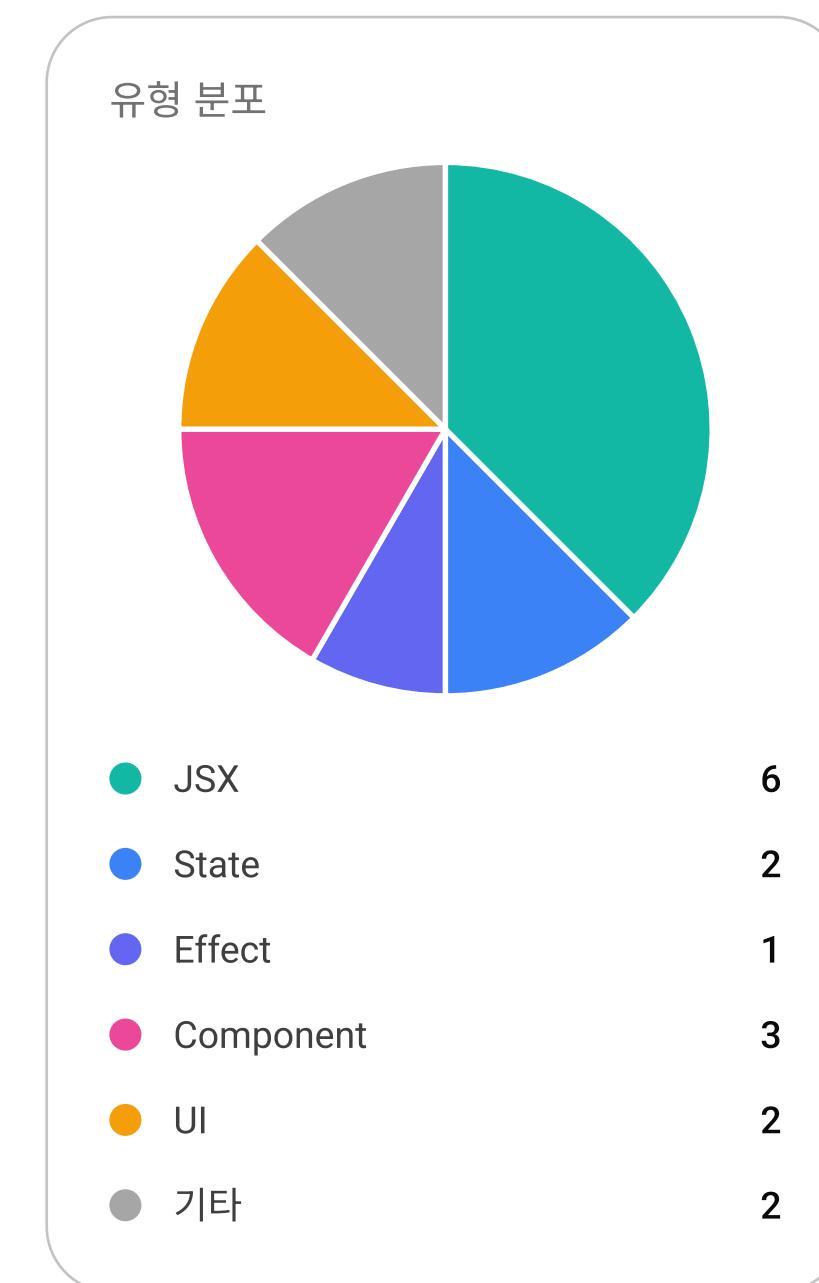
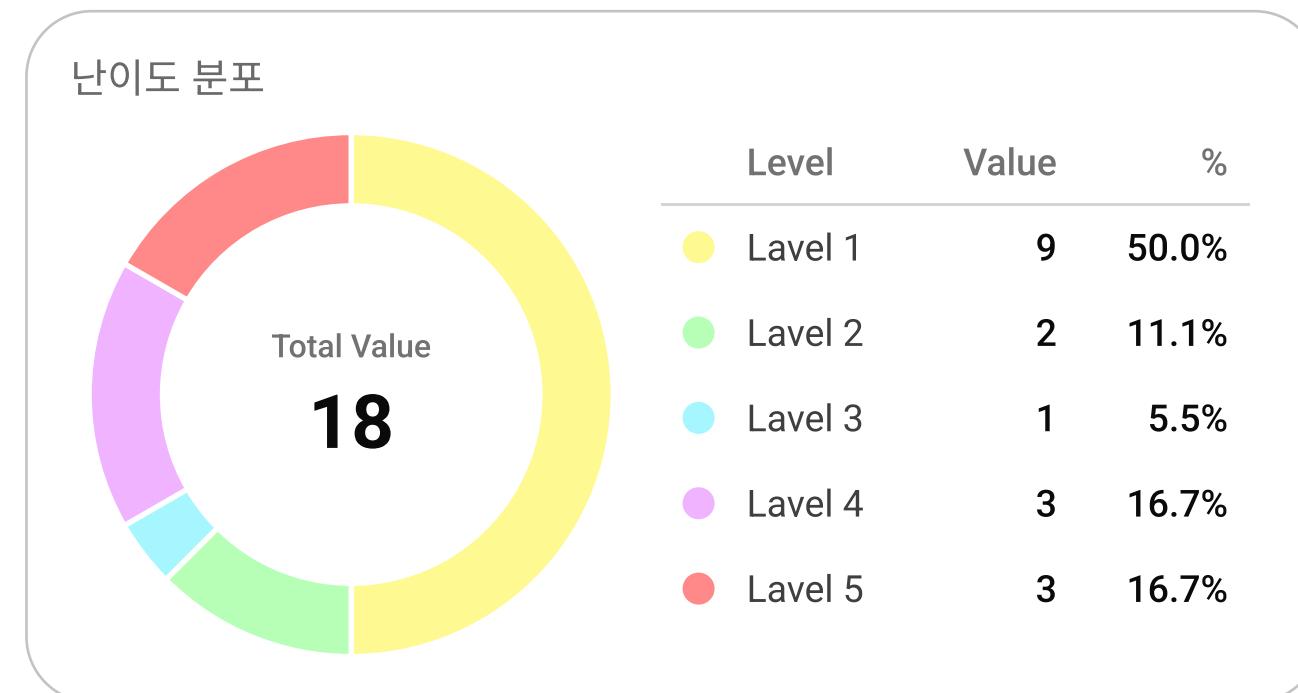
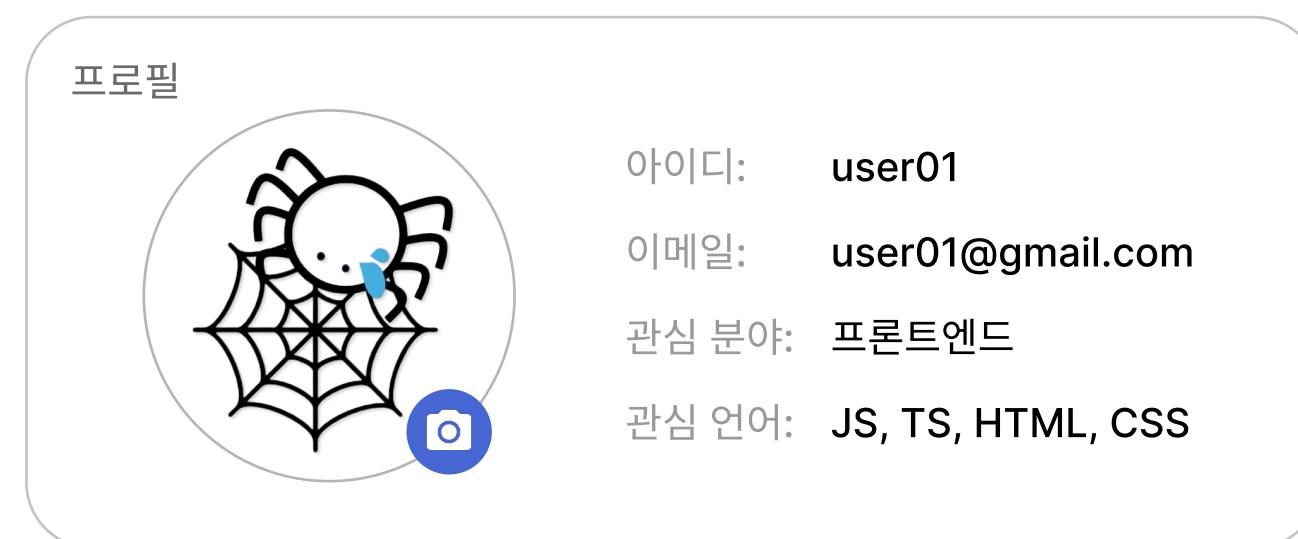
Level 3

다크 모드 및 동적 스타일링[UI](#) [UX](#)

학습 통계 문제



학습 통계 문제



학습 통계 문제**user01** 가 생성한 문제 5건

난이도	제목	정답률
Level 1	Button UI 생성하기	100.0%
Level 3	다크 모드 및 동적 스타일링	80.0%
Level 4	상태 관리와 버튼 연동	66.7%
Level 5	디자인 시스템 컴포넌트 통합	83.3%
Level 5	애니메이션과 접근성 향애니메이션과 접근성 향상	37.5%



문제 풀이

AI 챗봇

제출하기

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12}
13
```

Live Preview

Webshooter index #1

Webshooter index #2

Webshooter index #3



문제 풀이

AI 챗봇

제출하기

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12}
13
```

Live Preview

Webshooter index #1

Webshooter index #2

Webshooter index #3



문제 풀이

AI 챗봇

제출하기

문제 설명

/problem 엔드포인트에 **GET** 요청을 처리하는 API를 구현하세요. 이 API는 다양한 문제 목록을 담고 있는 배열을 **JSON** 형식으로 반환해야 합니다. 반환되는 문제들은 각 문제의 고유 ID, 제목, 난이도를 포함해야 합니다.

요구사항

1. 엔드포인트: /problem

- HTTP 메서드: GET
- 문제 목록 배열을 반환합니다. 각 문제는 고유 ID, 제목, 난이도를 포함해야 합니다.

2. 문제 데이터:

- 각问题是 다음의 속성들을 가져야 합니다:
 - id**: 고유한 문제 ID (정수형)
 - title**: 문제 제목 (문자열)
 - difficulty**: 문제 난이도 (문자열, 예: "쉬움", "중간", "어려움")

응답 예시

```
[  
  {  
    "id": 1,  
    "title": "간단한 API 구현",  
    "difficulty": "쉬움",  
  },  
  {  
    "id": 2,  
    "title": "데이터베이스 연결",  
    "difficulty": "중간",  
  },  
  {  
    "id": 3,
```

Problem.java ProblemController.java

```
3 import com.example.problemapi.model.Problem;  
4 import org.springframework.web.bind.annotation.GetMapping;  
5 import org.springframework.web.bind.annotation.RestController;  
6  
7 import java.util.Arrays;  
8 import java.util.List;  
9  
10 @RestController  
11 public class ProblemController {  
12  
13     @GetMapping("/problem")  
14     public List<Problem> getProblems() {  
15         return Arrays.asList(  
16             new Problem(1, "간단한 API 구현", "쉬움"),  
17             new Problem(2, "데이터베이스 연결", "중간"),  
18             new Problem(3, "로그인 기능 구현", "어려움")  
19         );  
20     }  
21 }
```

API Test Tool

GET {{ baseUrl }}/problem

Send

Query Body Headers Auth Vars

No Body

Response Headers

```
1 [  
2   {  
3     "id": 1,  
4     "title": "간단한 API 구현",  
5     "difficulty": "쉬움"  
6   },  
7   {  
8     "id": 2,  
9     "title": "데이터베이스 연결",  
10    "difficulty": "중간"  
11  },  
12  {  
13    "id": 3,  
14    "title": "로그인 기능 구현",  
15  }  
16 ]
```



문제 풀이

AI 챗봇

제출하기



안녕하세요, 무엇을 도와드릴까요?

추천 질문

컴포넌트가 무슨 개념이야?

Prop에 대해 설명해줘

어떤 점이 궁금하신가요?



App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12}
13
```

Live Preview

Webshooter index #1

Webshooter index #2

Webshooter index #3



문제 풀이

AI 챗봇

제출하기

Prop에 대해 설명해줘

React에서 **Prop** (Properties)은 컴포넌트 간에 데이터를 전달하는 방법입니다. React 컴포넌트는 재사용 가능한 UI 요소로 구성되며, 각 컴포넌트는 다른 컴포넌트로 데이터를 전달할 수 있습니다. 이때 사용되는 것이 props입니다.

Prop의 주요 개념

- 데이터 전달:** 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달할 때 사용됩니다. 이때 전달되는 데이터는 읽기 전용(read-only)이며, 자식 컴포넌트에서는 해당 데이터를 수정할 수 없습니다.
- 불변성:** props는 기본적으로 변경이 불가능합니다. 부모 컴포넌트에서 전달된 데이터는 자식 컴포넌트에서 읽기만 할 수 있고, 이를 직접 수정하려고 하면 React가 오류를 발생시킵니다. 자식 컴포넌트는 props를 받아서 화면을 렌더링하는 데만 사용합니다.
- 단방향 데이터 흐름:** React에서는 데이터가 항상 부모에서 자식으로 흐르며, 이를 통해 예측 가능한 상태를 유지할 수 있습니다. 즉, props는 부모 → 자식 방향으로만 전달됩니다.
- 기본값 설정:** 자식 컴포넌트에서 props를 받지 않으면 기본값을 설정할 수 있습니다. 이를 위해 defaultProps를 사용할 수 있습니다.

어떤 점이 궁금하신가요?



App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12 }
13 }
```

Live Preview

Webshooter index #1**Webshooter index #2****Webshooter index #3**



문제 풀이

AI 챗봇

제출하기

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12}
13 }
```

Live Preview Result

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 1: 통과

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100

공유하기



문제 풀이

AI 챗봇

제출하기

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 형태로 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {  
2   return <h2>Webshooter index #{index}</h2>;  
3 }  
4  
5 export default function ShooterSet() {  
6   return (  
7     <>  
8       <Shooter index={1} />  
9       <Shooter index={2} />  
10      <Shooter index={3} />  
11    </>  
12  )  
13}
```



문제를 공유하시겠습니까?

문제를 공유하면 다른 사용자들이 여러분의 제출 내역을 볼 수 있게 되고, 그에 대한 피드백을 주고받을 수 있습니다.

Level 3

Components와 Prop

prop Component

취소

확인

준: 95%)

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100

공유하기



문제 풀이

AI 챗봇

게시판

제출 ID	아이디	점수	제출 시각
#324	user13	100 / 100	2024-12-07 22:03

- user01 2024-12-08 11:23
함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?
- user03 2024-12-07 23:48
index 배열을 생성하고 map 함수를 사용하면 반복된 코드를 줄일 수 있을 것 같아요!
- user05 2024-12-07 22:32
저랑 풀이가 완전히 동일하네요!

user01

댓글을 작성해주세요. 불쾌감을 주는 욕설과 악플은 관리자에 의해 삭제될 수 있습니다.

댓글 등록

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  );
12 }
13 }
```

Live Preview Result

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 1: 통과

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100



문제 풀이

AI 챗봇

게시판

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     <>
8     <Shooter index={1} />
9     <Shooter index={2} />
10    <Shooter index={3} />
11  </>
12 );
13 }
```

Live Preview

Result

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 1: 통과

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100



문제 풀이

AI 챗봇

게시판

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받으세요.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #**index**"를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {  
2   return <h2>Webshooter index #{index}</h2>;  
3 }  
4  
5 export default function ShooterSet() {  
6   return (  
7     <div>
```

8번 line에 피드백을 추가

피드백을 작성해주세요. 불쾌감을 주는 욕설과 악플은 관리자에 의해 삭제될 수 있습니다.

취소

확인

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 1: 통과

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100



문제 풀이

AI 챗봇

게시판

문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1

Webshooter index #2

Webshooter index #3

App.js

```
1 function Shooter({ index }) {
2   return <h2>Webshooter index #{index}</h2>;
3 }
4
5 export default function ShooterSet() {
6   return (
7     ◇
8       <Shooter index={1} />
```

user03

2024-12-07 23:48

index 배열을 생성하고 map 함수를 사용하면 반복된 코드를 줄일 수 있을 것 같아요!

```
9   <Shooter index={2} />
10  <Shooter index={3} />
11  </>
12 );
13 }
```

Live Preview Result

테스트 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 1: 통과

테스트 2: 통과

테스트 3: 통과

테스트 4: 통과

채점 결과

UI 유사도 분석: 99.9% (정답 기준: 95%)

테스트 합계: 100/100

문제 사용자

전체 32건

[삭제](#)

	ID	난이도	제목	정답률
<input type="checkbox"/>	#1	Level 1	Button UI 생성하기	13
<input type="checkbox"/>	#2	Level 2	컴포넌트 생성	5
<input type="checkbox"/>	#3	Level 3	갤러리 에러 수정	3
<input type="checkbox"/>	#4	Level 4	Form Validation	2
<input type="checkbox"/>	#5	Level 5	채팅방 재연결 로직	4
<input type="checkbox"/>	#6	Level 3	재사용 가능한 컴포넌트	2
<input type="checkbox"/>	#7	Level 3	고급 이벤트 처리	0
<input type="checkbox"/>	#8	Level 3	다크 모드 및 동적 스타일링	0
<input type="checkbox"/>	#9	Level 4	상태 관리와 버튼 연동	0
<input type="checkbox"/>	#10	Level 4	Custom Hooks	0

문제

사용자

전체 32건

삭제

ID	난이도	제목	정답률
<input type="checkbox"/>	#1	Level 1 Button UI 생성하기	13
<input type="checkbox"/>	#2	Level 2 문제 삭제 확인	5
<input type="checkbox"/>	#3	Level 3 선택한 문제 • 고급 이벤트 처리 • 다크 모드 및 동적 스타일링	3
<input type="checkbox"/>	#4	Level 4 문제를 정말 삭제하시겠습니까?	2
<input type="checkbox"/>	#5	Level 5 삭제된 데이터는 복구할 수 없습니다.	4
<input type="checkbox"/>	#6	Level 3 취소 확인	2
<input checked="" type="checkbox"/>	#7	Level 3 고급 이벤트 처리	0
<input checked="" type="checkbox"/>	#8	Level 3 다크 모드 및 동적 스타일링	0
<input type="checkbox"/>	#9	Level 4 상태 관리와 버튼 연동	0
<input type="checkbox"/>	#10	Level 4 Custom Hooks	0



문제 설명

Shooter라는 컴포넌트를 생성하고, 이를 사용하여 여러 개의 **Shooter** 컴포넌트를 렌더링하는 **ShooterSet** 컴포넌트를 구현하세요. 각 **Shooter** 컴포넌트는 **prop**으로 **index** 값을 받아 해당 값을 표시합니다.

요구사항

1. **Shooter**라는 이름의 컴포넌트를 생성하세요.
2. **Shooter** 컴포넌트는 하나의 **prop**으로 **index**를 받아야 합니다.
3. **Shooter** 컴포넌트는 **h2** 태그로 "Webshooter index #{index}"라는 텍스트를 렌더링해야 합니다.
 - 예: **index** 값이 1일 때는 "Webshooter index #1"이 표시되어야 합니다.
4. **ShooterSet**이라는 이름의 컴포넌트를 생성하여, 이 컴포넌트에서 **Shooter** 컴포넌트를 3번 호출하세요.
 - 각각의 호출에서는 **index** 값을 1, 2, 3으로 넘겨야 합니다.
5. **ShooterSet** 컴포넌트는 3개의 **Shooter** 컴포넌트를 렌더링해야 합니다.

화면 예시

Webshooter index #1
Webshooter index #2
Webshooter index #3

App.js

```
1 function Shooter({ index }) {  
2   return <h2>Webshooter index #{index}</h2>;  
3 }  
4  
5 export default function ShooterSet() {  
6   return (  
7     <>  
8       <Shooter index={1} />  
9       <Shooter index={2} />  
10      <Shooter index={3} />  
11    </>  
12  );  
13}
```

Live Preview

Webshooter index #1

Webshooter index #2

Webshooter index #3

[문제](#) [사용자](#)

전체 43건

[삭제](#)

	ID	아이디	권한	생성한 문제 수	생성한 댓글 수
<input type="checkbox"/>	#1	admin	관리자	13	0
<input type="checkbox"/>	#2	user01	일반 사용자	5	4
<input type="checkbox"/>	#3	user02	일반 사용자	3	5
<input type="checkbox"/>	#4	user03	일반 사용자	2	3
<input type="checkbox"/>	#5	user04	일반 사용자	4	2
<input type="checkbox"/>	#6	user05	일반 사용자	2	1
<input type="checkbox"/>	#7	user06	일반 사용자	0	0
<input type="checkbox"/>	#8	user07	일반 사용자	0	0
<input type="checkbox"/>	#9	user08	일반 사용자	0	0
<input type="checkbox"/>	#10	user09	일반 사용자	0	0



문제

사용자

전체 43건

[삭제](#)

	ID	아이디	권한	생성한 문제 수	생성한 댓글 수
<input type="checkbox"/>	#1	admin	관리자	13	0
<input type="checkbox"/>	#2	user01	사용자 삭제 확인	4	5
<input type="checkbox"/>	#3	user02			
<input type="checkbox"/>	#4	user03	선택한 사용자 user08과 user09를 정말 삭제하시겠습니까?	3	2
<input type="checkbox"/>	#5	user04			
<input type="checkbox"/>	#6	user05	삭제된 데이터는 복구할 수 없습니다.	1	0
<input type="checkbox"/>	#7	user06			
<input type="checkbox"/>	#8	user07	일반 사용자	0	0
<input checked="" type="checkbox"/>	#9	user08	일반 사용자	0	0
<input checked="" type="checkbox"/>	#10	user09	일반 사용자	0	0

[취소](#)[확인](#)



문제

사용자

프로필



아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드
관심 언어: JS, TS, HTML, CSS

작성한 댓글 / 피드백 3건

삭제

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이...	피드백
<input type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

[문제](#) [사용자](#)

프로필



아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드
관심 언어: JS, TS, HTML, CSS

작성한 댓글 / 피드백 3건

[삭제](#)

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글 
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이 컴포넌트의 렌더링과 관련이 없다면 useRef를 사용하는 것 이 더 적합합니다! useRef는 값이 변경되어도 컴포넌트를 재렌더링하지 않기 때문에, 렌더링에 영향을 주지 않으면서 도 값을 유지할 수 있습니다!	피드백 
<input type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백 

문제 사용자

프로필



아이디: user01
이메일: user01@gmail.com
관심 분야: 프론트엔드
관심

작성한 댓글 / 피드백 3건

댓글 ID	문제 ID	내용	종류
<input type="checkbox"/> #37	#15	함수 선언문 대신에 함수 표현식을 쓰는 건 어떨까요?	댓글
<input type="checkbox"/> #43	#18	현재 코드에서 useState를 사용하고 있는데. 해당 값이...	피드백
<input checked="" type="checkbox"/> #47	#18	이 코드는 없어도 똑같이 작동할 것 같아요!	피드백

댓글 삭제 확인

X

선택한 피드백#47를 정말 삭제하시겠습니까?
삭제된 데이터는 복구할 수 없습니다.