



책스터디 [모던 리액트 Deep Dive]



문제	답변	날짜	Person	범위
내일 범위	14. 웹사이트 보안을 위한 리액트와 웹페이지 보안 이슈 (p. 875) - 15. 마치며 (p. 916)	@2024년 1월 19일 오후 2:00	지찬 주 JIN	
공지	1월 15일 월요일 - (시간 변경) 오후 7시로! 1월 16일 화요일 - 오후 4시 삼성역 가배도 1월 19일 ~ 1월 20일 - 휴			
리액트를 사용할 경우 장점	1. 자바스크립트 코드의 크기가 줄었다. 상태를 관리하기 위한 컨트roller 대신 리액트는 단지 상태에 따른 ui를 선언적으로 구현 가능하기 때문. 2. 상대적으로 완전한 학습곡선, 자바스크립트코드와 html만 알면 손쉽게 코드작성 가능 3. 빠른 기능추가 : 빌드하는 시간이 훨씬 빠르다	@2024년 1월 2일	지찬 주	
typeof null 했을때의 반환값은?	Object이며, 이는 JavaScript의 초기 버전에서의 설계 결함 중 하나로 여겨진다.	@2024년 1월 2일	지찬 주	
BigInt의 범위는?	$-2^{53} - 1 \sim 2^{53} - 1$ Number.MAX_SAFE_INTEGER ~ Number.MIN_SAFE_INTEGER = 9007199254740991 ~ -9007199254740991	@2024년 1월 2일	지찬 주	
원시타입과 객체타입의 저장 형태는?	원시타입은 불변의형태로 저장, 객체타입은 변경가능한형태로 저장	@2024년 1월 2일	지찬 주	
호이스팅이란 ?	함수에대한 선언을 실행전에 미리 메모리에 등록하는 작업	@2024년 1월 2일	지찬 주	
리액트에서는 동등 비교를 어떻게 하는가?	Object.is 기반으로 shallowEqual 함수를 만들어 사용한다. 원시값 비교 (타입까지 비교) 는 Object.is로 하고 Object.is로 하지 못하는 부분, 즉 객체 간 비교 depth 1까지 한다.	@2024년 1월 2일	JIN	
화살표 함수와 일반 함수의 차이점	this 바인딩에서 가장 큰 차이를 보인다. 화살표 함수의 경우 상위 스코프의 this를 그대로 따르게 되고 일반 함수는 함수가 호출되는 방식에 따라 달라진다.	@2024년 1월 2일	JIN	

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
<u>함수 만들때 주의하면 좋은 점</u>	함수의 부수 효과를 최대한 억제하라, 가능한 한 함수를 작게 만들어라, 누구나 이해할 수 있는 이름을 붙여라	@2024년 1월 2일	J JIN	
<u>클로저란 무엇이고 장점과 단점은?</u>	함수와 함수가 선언된 어휘적 환경의 조합이며, 장점은 개발자가 원하는 정보만 원하는 방향으로 노출시킬 수 있고, 단점은 성능에 악영향을 끼칠 가능성이 있다. 내부함수는 외부함수의 선언적 환경을 기억하고있어야 하기때문	@2024년 1월 3일	지찬 주	
<u>자바스크립트는 싱글 스레드 언어인데 비동기 프로세스처리는 어떻게 하는가?</u>	태스크 큐가 할당되는 별도의 스레드에서 수행된다.	@2024년 1월 3일	지찬 주	
<u>이벤트 루프란?</u>	자바스크립트 런타임 외부에서 비동기 실행을 돕기 위해 만들어진 장치	@2024년 1월 3일	지찬 주	
<u>일급객체란?</u>	다른 객체들에 일반적으로 적용 가능한 연산을 모두 지원하는 객체	@2024년 1월 2일	지찬 주	
<u>클래스에 인수로 넘겨준 변수의 prototype을 확인할 수 있는 방법은 ?</u>	Object.getPrototypeOf()	@2024년 1월 3일	지찬 주	
<u>이벤트루프가 호출스택에서 하는 역할</u>	이벤트처리 및 비동기 작업 관리, 호출 스택이 비어있는지 확인, 비어있다면 수행해야할 코드가 있는지 확인하고 호출 스택에 추가	@2024년 1월 3일	지찬 주	
<u>렉시컬 환경이란 무엇인가요? 동적으로 결정되나요 정적으로 결정되나요?</u>	변수가 코드 내부에서 어디서 선언됐는지를 말하는 것, 정적	@2024년 1월 3일	J JIN	
<u>자바 스크립트가 멀티 스레드로 구현되지 않은 이유는 뭐라고 생각하시나요?</u>	당시에는 이렇게 많은 일을 수행하게 될 줄 몰랐고, 멀티 스레드의 단점을 생각해봤을 때 여러 문제점을 야기했을 것이다.	@2024년 1월 3일	J JIN	
<u>태스크 큐와 마이크로 큐 대표 작업 예시를 들어주세요</u>	<ul style="list-style-type: none"> • 태스크 큐 : setTimeout • 마이크로 큐 : Promise 	@2024년 1월 3일	J JIN	
<u>JSX요소명이 대문자로 시작하는 이유는 ?</u>	html태그명과 사용자가 만든 컴포넌트 태그명을 구분짓기위해	@2024년 1월 4일	지찬 주	
<u>DOM 이란?</u>	웹페이지에 대한 인터페이스로 브라우저가 웹페이지의 콘텐츠와 구조를 어떻게 보여줄지에 대한 정보를 담고있다.	@2024년 1월 4일	지찬 주	
<u>브라우저 렌더링 과정은 ?</u>	• 브라우저가 사용자가 요청한 주소를 방문해 HTML 파일을 다운로드한다.	@2024년 1월 4일	J JIN	

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
	<ul style="list-style-type: none"> • 브라우저의 렌더링 엔진은 HTML을 파싱해 DOM 노드로 구성된 트리(DOM)를 만든다. • 2번 과정에서 CSS 파일을 만나면 해당 CSS 파일도 다운로드한다. • 브라우저의 렌더링 엔진은 이 CSS도 파싱해 CSS 노드로 구성된 트리(CSSOM)를 만든다. • 브라우저는 2번에서 만든 DOM 노드를 순회하는데. 여기서 모든 노드를 방문하는 것이 아니고 사용자 눈에 보이는 노드만 방문한다. 즉, display: none과 같이 사용자 화면에 보이지 않는 요소는 방문하지 않는다. 이는 트리를 분석하는 과정을 조금이라도 빠르게 하기 위해서다. • 5번에서 제외된, 눈에 보이는 노드를 대상으로 해당 노드에 대한 CSSOM 정보를 찾고 여기서 발견한 CSS 스타일 정보를 이 노드에 적용한다. <p>레이아웃 : 각 노드가 브라우저 화면의 어느 좌표에 정확히 나타나야 하는지 계산하는 과정. 이 레이아웃 과정을 거치면 반드시 페인팅 과정도 거치게 된다.</p> <p>페인팅: 레이아웃 단계를 거친 노드에 색과 같은 실제 유효한 모습을 그리는 과정.</p>			
<u>JSX가 반드시 트랜스파일러를 거쳐야 하는 이유는?</u>	자바스크립트 표준코드가 아닌 페이스북이 임의로 만든 새로운 문법이기때문에	@2024년 1월 4일	 지찬 주	
<u>타입을 최대한 좁히는 것에 도움을 주는행위는?</u>	타입가드(instanceof, typeof)	@2024년 1월 4일	 지찬 주	
<u>리액트 파이버란?</u>	가상 DOM과 실제 DOM을 비교해 변경사항을 수집하며 만약 이 둘 사이에 차이가 있다면 변경된 정보를 가지고 있는 파이버를 기준으로 화면에 렌더링을 요청하는 역할을 한다	@2024년 1월 4일	 지찬 주	
<u>가상 DOM의 등장 배경은?</u>	렌더링 이후에 사용자 인터랙션을 통해 웹페이지가 계속해서 변경이 되는데 그때마다 브라우저 렌더링 과정 중 레이아웃과 리페인팅이 계속 발생하며 더 많은 비용으로 이어지고, 개발자 입장에서도 다양한 변경 사항을 추적하는 것도 수고스러운 일이었다. 그래서 최종 결과물 하나만 알고 싶었기 때문에 가	@2024년 1월 4일	 JIN	

📄 문제	✔️ 답변	📅 날짜	👤 Person	≡ 범위
	상 DOM 을 활용하여 메모리에 저장하고 실제 변경에 대한 준비가 끝나면 실제 브라우저 DOM에 반영한다. 메모리에서 계산하는 과정을 거침으로 렌더링 과정을 최소화 하고 브라우저와 개발자 입장에서 부담이 덜어진다.			
<u>리액트 요소와 파이버의 차이점은?</u>	리액트 요소는 렌더링이 발생할 때마다 새롭게 생성, 파이버는 가급적 재사용 (재사용 하는 이유 : 변경이 일어날때마다 재생성하기에는 변경이 잦기에 오히려 낭비가 될 수 있기 때문에 기존 객체를 재 활용 한다.)	@2024년 1월 4일	J JIN	
<u>클래스형 컴포넌트에서, React.Component와 React.PureComponent의 차이는?</u>	<ul style="list-style-type: none"> • Component는 state가 업데이트 되는대로 렌더링되지만 PureComponent는 얕은비교를 수행해 결과가 다를 때만 렌더링을 수행한다. 	@2024년 1월 5일	지찬 주	2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98) - 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모이제이션 (p.188)
<u>클래스형 컴포넌트의 생명주기 메서드가 실행되는 시점은?</u>	<ul style="list-style-type: none"> • 마운트 : 컴포넌트가 마운팅 되는 시점 • 업데이트 : 이미 생성된 컴포넌트의 내용이 변경되는 시점 • 언마운트 : 컴포넌트가 더이상 존재하지 않는 시점 	@2024년 1월 5일	지찬 주	2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98) - 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모이제이션 (p.188)






📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>클래스형 컴포넌트와 함수형 컴포넌트의 차이점</u>	<ul style="list-style-type: none"> • 클래스형 컴포넌트의 생명주기 메서드가 함수형 컴포넌트에는 존재하지 않는다 : 함수형 컴포넌트는 props를 받아 단순히 리액트 요소만 반환하는 함수인 반면, 클래스형 컴포넌트는 render 메서드가 있는 React.Component를 상속 받아 구현하는 자바스크립트 클래스이기 때문 반면 함수형 컴포넌트는 useEffect 혹은 통해 클래스형 컴포넌트의 생명주기 메서드를 비슷하게 구현할 수 있다. 	@2024년 1월 5일	 지찬 주	2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98) - 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모이제이션 (p.188)
<u>리액트의 렌더링은 언제 발생하는가?</u>	<ol style="list-style-type: none"> 1. 최초렌더링 2. 리렌더링 시 	@2024년 1월 5일	 지찬 주	2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98) - 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모이제이션 (p.188)
<u>리액트의 리렌더링은 언제 발생하는가?</u>	<ul style="list-style-type: none"> • 클래스형 컴포넌트의 setState가 실행되는 경우 • 클래스형 컴포넌트의 forceUpdate가 실행되는 경우 • 함수형 컴포넌트의 useState()의 두 번째 배열 요소인 setter가 실행되는 경우 • 함수형 컴포넌트의 useReducer()의 두 번째 요소인 dispatch가 실행되는 경우 • 컴포넌트의 key props가 변경되는 경우 	@2024년 1월 5일	 지찬 주	2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98) - 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				이제이션 (p.188)
<p><u>클래스형 컴포넌트의 한계점은?</u></p>	<ol style="list-style-type: none"> 1. 데이터의 흐름을 추적하기 어렵다 : 메서드의 순서를 안맞춰줘도 되기때문에 state의 흐름을 읽기 힘들다 2. 애플리케이션 내부 로직의 재사용이 어렵다. : 공통로직이 많아질수록 이를 감싸는 컴포넌트 내지는 props가 많아지는 래퍼지옥에 빠져들 위험성이 커질 수 있고, 재사용할 로직도 많아지는데 이를 클래스형 컴포넌트 환경에서 매끄럽게 처리하기 쉽지 않다. 3. 기능이 많아질수록 컴포넌트의 크기가 커진다. : 로직이 많아질수록 생명주기 메서드 사용이 잦아져 컴포넌트의 크기가 기하급수적으로 커진다. 4. 클래스는 함수에 비해 상대적으로 어렵다. : 자바스크립트 개발자는 클래스 보다는 함수에 더 익숙하다. 5. 코드 크기를 최적화하기 어렵다. : 빌드 시 사용되지 않는 메서드도 그대로 번들된다. 6. 핫 리로딩을 하는 데 상대적으로 불리하다. : 클래스형 컴포넌트는 최초 렌더링 시에 instance를 생성하고, 그 내부에서 state 값을 관리하는데, 이 instance 내부에 있는 render를 수정하게 되면 이를 반영할 수 있는 방법은 오직 instance를 새로 만드는 것뿐이다. 그리고 새롭게 만들어진 instance에서 값은 당연히 초기화될 수밖에 없다. 이에 반해 함수형 컴포넌트를 state를 함수가 아닌 클로저에서 저장해 두므로 함수가 다시 실행돼도 해당 state를 잃지 않고 다시 보여줄 수 있게 된다. 	@2024년 1월 5일	 지찬 주	<p>2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98)</p> <p>- 2.5 컴포넌트와 함수의 무거운 연산을 기억해두는 메모이제이션 (p.188)</p>
<p><u>메모이제이션에 대해서 자신의 의견을 말해주세요</u></p>	<ul style="list-style-type: none"> • 선택된 메모이제이션을 지양하고, 성능에 어떠한 영향을 미치는지 살펴 본 후 적용할 것이다. 	@2024년 1월 5일	 지찬 주	<p>2.3 클래스형 컴포넌트와 함수형 컴포넌트는 (p.98)</p> <p>- 2.5 컴포넌트와 함수</p>

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				수의 무 건운 연 산을 기 억해두 는 메모 이제이 션 (p.188)
<u>리액트 렌더링과 브라우 저 렌더링 차이</u>	<ul style="list-style-type: none"> • 브라우저 렌더링은 HTML CSS 기반으로 사용자에게 보여줄 화면을 그리는 작업 • 리액트 렌더링은 리액트 애플리케이션 트리 안에 있는 모든 컴포넌트들이 인자와 상태값을 기반으로 어떻게 화면을 구성하고 DOM 결과를 브라우저에 제공할지 계산하는 과정 	@2024년 1월 5일	 JIN	2.3 클 래스형 컴포넌 트와 함 수형 컴 포너는 (p.98) - 2.5 컴포넌 트와 함 수의 무 건운 연 산을 기 억해두 는 메모 이제이 션 (p.188)
<u>리액트 렌더링의 렌더 단 계와 커밋 단계</u>	차이가 있음을 구분하는 단계가 렌더 단계, 차이가 있는 경우 화면에 적용하는 것이 커밋 단계	@2024년 1월 5일	 JIN	2.3 클 래스형 컴포넌 트와 함 수형 컴 포너는 (p.98) - 2.5 컴포넌 트와 함 수의 무 건운 연 산을 기 억해두 는 메모 이제이 션 (p.188)
<u>즉시 실행 함수의 장점</u>	한번만 실행되고 재사용이 안되기 때문에 글로벌 스코프를 오염시키지 않으며 독립적이어서 리팩터링 때도 직관적이다.	@2024년 1월 5일	 JIN	
<u>케이크 초기화는 어떻게 사용하나요?</u>	<ul style="list-style-type: none"> • useState의 초기값이 복잡하거나 무거운 연산을 포함하고 있을 때, 	@2024년 1월 6일	 지찬 주	3.1 리 액트의 모든 혹

📄 문제	✔️ 답변	📅 날짜	👤 Person	≡ 범위
				파헤치기 (p. 189 - p.251)
<u>useState의 게으른 초기화(기본값을 함수로 두는 경우⇒state가 처음 만들어질 때만 사용되고 그 뒤로는 무시된다.)를 사용하는 경우는 어떤 경우인가요?</u>	<ul style="list-style-type: none"> 초기값이 복잡하거나 무거운 연산을 수행하는 경우 localStorage, sessionStorage에 접근하는 경우, map, filter, find 메서드 같이 배열에 대한 접근, 초기값 계산을 위해 함수 접근이 필요할 때 	@2024년 1월 6일	J JIN	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useEffect의 두번째 인자에 아무것도 넣지 않으면 매 렌더링 모습을 알 수 있다. useEffect가 아니어도 해당 동작은 가능한데 두번째 인자가 없는 useEffect만의 동작 특징이 있는가?</u>	<ul style="list-style-type: none"> 클라이언트 사이드에서 실행되는 것을 보장 useEffect는 컴포넌트 렌더링 이후에 발생한다. 반면 직접 실행은 컴포넌트 렌더링 도중에 실행된다. (서버사이드의 경우 서버에서도 실행) 그래서 직접 실행은 함수형 컴포넌트 반환을 지연시키는 행위이다. 	@2024년 1월 6일	J JIN	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useContext 혹은 상태 관리 혹은이라고 할 수 없는 이유는</u>	상태를 주입할 뿐 상태를 변경하거나 렌더링 최적화에 영향을 주는 기능을 수행하지 않기 때문	@2024년 1월 6일	J JIN	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useState에 대해서 설명해주세요</u>	<ul style="list-style-type: none"> 함수형 컴포넌트 내부에서 상태를 정의하고, 이 상태를 관리할 수 있게 해주는 훅 	@2024년 1월 6일	지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useState는 어떻게 state의 값을 유지하고 있나요?</u>	<ul style="list-style-type: none"> state의 값을 유지하고 사용하기 위해서 리액트는 클로저를 활용하고 있다. 	@2024년 1월 6일	지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useEffect의 클린업 함수란?</u>	<ul style="list-style-type: none"> useEffect의 콜백 함수가 이벤트를 등록했을때, 그 것을 지울 때 사용하는 것 useEffect는 그 콜백이 실행될 때마다, 이전의 클린업 함수가 존재한다면 그 클린업 함수를 실행한 뒤에 콜백을 실행한다. 	@2024년 1월 6일	지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>useEffect는 의존성배열에 값을 어떻게 비교하는가?</u>	<ul style="list-style-type: none"> Object.is를 기반으로 하는 얇은 비교 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useEffect에서 의존성 배열에 빈 배열을 지양하는 이유</u>	<ul style="list-style-type: none"> useEffect는 반드시 의존성 배열로 전달한 값의 변경에 의해 실행돼야 하는 훅이다, 그러나 의존성 배열을 넘기지 않은 채 콜백 함수 내부에서 특정 값을 사용한다는 것은, 이 부수 효과가 실제로 관찰해서 실행돼야 하는 값과는 별개로 작동한다는 것을 의미한다. 즉, 컴포넌트의 state, props와 같은 어떤 값의 변경과 useEffect의 부수 효과가 별개로 작동하게 된다는 것이다. useEffect에서 사용한 콜백 함수의 실행과 내부에서 사용한 값의 실제 변경 사이에 연결 고리가 끊어져 있는 것이다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useEffect 콜백 인수로 비동기함수를 지정 할 수 없는 이유</u>	<ul style="list-style-type: none"> 경쟁상태가 발생할 수 있다. 비동기 함수를 지정할 수 없는 것이, 비동기 함수 실행 자체가 문제가 되는것은 아니다, 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useMemo란?</u>	<ul style="list-style-type: none"> 비용이 큰 연산에 대한 결과를 저장해 두고, 이 저장된 값을 반환하는 훅 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useCallback이란?</u>	<ul style="list-style-type: none"> 인수로 넘겨받은 콜백 자체를 기억하는것, 쉽게 말해 함수를 메모이제이션한다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useMemo와 useCallback 둘의 차이점은?</u>	<ul style="list-style-type: none"> useMemo는 변수를 메모이제이션하고, useCallback은 함수를 메모이제이션한다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
<u>useRef는 무엇이고, useState와 구별되는 큰 차이점은?</u>	<ul style="list-style-type: none"> • 컴포넌트 내부에서 렌더링이 일어나도 변경 가능한 상태값을 저장하는 것 • 차이점 1. useRef는 반환값인, 객체 내부에 있는 current로 값에 접근 또는 변경할 수 있다. • 차이점 2. useRef는 그 값이 변하더라도 렌더링을 발생시키지 않는다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useRef의 장점</u>	<ul style="list-style-type: none"> • 컴포넌트가 렌더링 될때만 생성되는 변수이다. • 컴포넌트 인스턴스가 여러 개라도 각각 별개의 값을 바라볼 수 있다. • 개발자가 원하는 시점의 값을 렌더링에 영향을 미치지 않고 보관할 수 있다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useContext란?</u>	<ul style="list-style-type: none"> • props drilling을 해결하기 위해 등장했다. • 명시적인 props 전달 없이도 선언한 하위 컴포넌트 모두에서 자유롭게 원하는 값을 사용할 수 있다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useContext를 사용할 때 주의할 점</u>	<ul style="list-style-type: none"> • 컴포넌트 재활용이 어려워진다, useContext가 선언돼 있으면 Provider에 의존성을 가지고 있는 셈이 되므로 재활용 하기 어려워진다. • 모든 컨텍스트를 최상위 루트 컴포넌트에 넣으면 안된다, 컨텍스트가 많아질수록 루트 컴포넌트는 더 많은 컨텍스트로 둘러싸일 것이고, 해당 props를 다수의 컴포넌트에서 사용할 수 있게끔 해야 하므로 불필요하게 리소스가 낭비된다. 따라서 컨텍스트가 미치는 범위는 필요한 환경에서 최대한 좁게 만들어야 한다. • 상태 관리를 위한 리액트 API가 아니다, 상태관리 API조건은 <ol style="list-style-type: none"> 1. 어떠한 상태를 기반으로 다른 상태를 만들어낼 수 있어야 한다 2. 필요에 따라 이러한 상태 변화를 최적화할 수 있어야 한다. 3. 컨텍스트는 둘 중 어느것도 하지 못한다. 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 혹은 파헤치기 (p. 189 - p.251)
<u>useReducer란?</u>	<ul style="list-style-type: none"> • state값을 변경하는 시나리오를 제한적으로 두고 이에 대한 변경을 빠르게 확인할 수 있게끔 하는 	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 혹은 파헤치

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
				기 (p. 189 - p.251)
<u>useState와 useReducer의 공통점과 차이점</u>	공통점 : 클로저를 활용해 값을 가둬서 state를 관리한다 차이점 : 세부 작동과 쓰임에만 차이가 있다.	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>forwardRef란?</u>	• ref를 전달하는 데 있어서 일관성을 제공하기 위해, props에 ref 약어를 사용하면 에러발생	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useImperativeHandle이란?</u>	• 부모에게서 넘겨받은 ref를 원하는 대로 수정할 수 있는 훅	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>useLayoutEffect는 무엇이고 useEffect와 공통점과 차이점은?</u>	공통점 : useEffect와 사용의 형태가 별반 다르지 않다. 차이점 : useLayoutEffect는 모든 리액트 DOM의 변경 후에 실행된다. useEffect는 브라우저에 변경 사항이 반영된 이후에 실행된다	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>훅의 규칙 두가지</u>	1. 최상위에서만 훅을 호출해야 한다. 반복문이나 조건문, 중첩된 함수 내에서 훅을 실행할 수 없다. 이 규칙을 따라야만 컴포넌트가 렌더링 될 때마다 항상 동일한 순서로 훅이 호출되는 것을 보장할 수 있다. 2. 훅을 호출할 수 있는 것은 리액트 함수형 컴포넌트, 혹은 사용자 정의 훅의 두 가지 경우뿐이다. 일반 자바스크립트 함수에서는 훅을 사용할 수 없다.	@2024년 1월 6일	 지찬 주	3.1 리액트의 모든 훅 파헤치기 (p. 189 - p.251)
<u>사용자 정의 훅과 고차 컴포넌트의 차이점</u>	• 공통점 : 여러 컴포넌트에 반복되는 로직이면 따로 분리하여 컴포넌트 크기를 줄이고 가독성을 향상시킨다. • 사용자 정의 훅 : 리액트에서만 사용 가능, use로 시작, 렌더링 결과물에는 영향을 주지 않기에 컴포	@2024년 1월 8일	 JIN	3.2 사용자 정의 훅과 고차 컴포넌트 중 무엇을 써야

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
	<p>넌트 내부에 미치는 영향을 최소화해 개발자가 원하는 방향으로 사용이 가능하다.</p> <ul style="list-style-type: none"> 고차 컴포넌트 : 고차 함수의 일종으로 자바스크립트 환경에서 사용 가능, with로 시작, 렌더링 결과물에 영향을 주는 로직의 경우일 때 많이 사용하므로 부수 효과를 최소한으로 한다던지 컴포넌트에 미치는 영향을 좀 더 고려해야 한다. 			<p>할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)</p>
<p><u>JAM 스택은 어떤 문제를 해결했나요?</u></p>	<p>서버에 의존적인데 서버 확장에 어려움을 겪는 문제를 해결, 자바스크립트와 마크업을 미리 빌드해놓고 정적으로 사용자에게 제공하면 작동은 클라이언트에서 실행</p>	@2024년 1월 8일	J JIN	<p>3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)</p>
<p><u>싱글 페이지 어플리케이션과 서버 사이드 렌더링의 차이는?</u></p>	<p>웹페이지 렌더링을 어디서 하는가, 클라이언트에서 한다 ⇒ SPA, 서버 사이트라면 서버에서 제공하여 사용자 기기 성능 영향을 덜 받는다.</p>	@2024년 1월 8일	J JIN	<p>3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)</p>

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>서버 사이드 렌더링이 검색엔진에 좀 더 최적화된 이유?</u>	검색엔진 로봇이 HTML 을 다운로드만 하고 JS 실행을 하지 않기 때문에, SPA 는 작동 대부분이 JS 의존적인데 검색 엔진에 저장되는 메타정보 또한 그렇다. 페이지 최초 진입 시 메타 정보를 제공할 수 있도록 조치를 취하지 않으면 검색엔진, SNS 공유에 불이익이 있을 수 있다. 반면 SSR은 렌더링 작업이 서버에서 일어나므로 검색엔진에 제공할 정보를 서버에서 가공하여 HTML 응답으로 제공할 수 있다.	@2024년 1월 8일	J JIN	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>서버 사이드 렌더링을 위한 API 중 renderToNodeStream API는 왜 필요한가?</u>	생성해야하는 HTML 크기가 매우 클 때 서버에 무리가 될 수 있기 때문에 청크 단위로 분리해 순차적으로 처리할 수 있는 장점이 있다.	@2024년 1월 8일	J JIN	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>사용자 정의 훅이란 ?</u>	• 서로 다른 컴포넌트 내부에서 같은 로직을 공유하고자 할 때 주로 사용되는 것이며 리액트에서만 사용할 수 있다.	@2024년 1월 8일	지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				위한 리액트 API 살펴보기 (p. 293)
<u>고차 컴포넌트란?</u>	<ul style="list-style-type: none"> 컴포넌트 자체의 로직을 재사용하기 위한 방법이며 굳이 리액트가 아니더라도 자바스크립트 환경에서 널리 쓰일 수 있다. 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 훅과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>고차 함수란?</u>	<ul style="list-style-type: none"> 함수를 인수로 받거나 결과로 반환하는 함수 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 훅과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>사용자 정의 훅이 필요한 경우는 어떤 때인가?</u>	<ul style="list-style-type: none"> 단순히 useEffect, useState와 같이 리액트에서 제공하는 훅으로만 공통 로직을 격리할 수 있다면 사용자 정의 훅을 사용하는 것이 좋다. 사용자 정의 훅은 그 자체로는 렌더링에 영향을 미치지 못하기 때문에 사용이 제한적이므로 반환 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 훅과 고차 컴포넌트 중 무엇을 써야

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
	하는 값을 바탕으로 무엇을 할지는 개발자에게 달려 있다. 따라서 컴포넌트 내부에 미치는 영향을 최소화해 개발자가 혹은 원하는 방향으로만 사용할 수 있다는 장점이 있다.			할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>고차 컴포넌트를 사용해야 하는 경우는 어떤 때인가?</u>	<ul style="list-style-type: none"> 함수형 컴포넌트의 반환값, 즉 렌더링의 결과물에도 영향을 미치는 공통 로직일때, 즉 공통화된 렌더링 로직을 처리할때, 하지만 고차 컴포넌트가 많아질수록 복잡성이 기하급수적으로 증가하므로 신중하게 사용해야 한다. 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>싱글 페이지 어플리케이션이란?</u>	<ul style="list-style-type: none"> 렌더링과 라우팅에 필요한 대부분의 기능을 서버가 아닌 브라우저의 자바스크립트에 의존하는 방식 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>JAM 스택이란?</u>	<ul style="list-style-type: none"> • Javascript, API, Markup 스택의 약자 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>서버 사이드 렌더링이란?</u>	<ul style="list-style-type: none"> • 사용자에게 보여줄 페이지를 서버에서 렌더링해 빠르게 사용자에게 화면을 제공하는 방식 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>서버 사이드 렌더링의 장점은?</u>	<ul style="list-style-type: none"> • 최초 페이지 진입이 비교적 빠르다 :-> 일반적으로 서버에서 HTTP 요청을 수행하는 것이 더 빠르다 하지만 충분한 리소스가 확보돼 있지 않다면 오히려 SPA보다 느려질 수 있다. • 검색 엔진과 SNS 공유 등 메타데이터 제공이 쉽다 :-> 검색 엔진 최적화에 유용하다 검색 엔진은 페이지의 정적인 정보를 가져오는 것이 목적이다. • 검색 엔진은 어떻게 작동하는가? : 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을

문제	답변	날짜	Person	범위
	<p>1. 검색 엔진 로봇이 페이지에 진입한다.</p> <p>2. 페이지가 HTML 정보를 제공해 로봇이 이 HTML을 다운로드 한다. 단, 다운로드만 하고 자바스크립트 코드는 실행하지 않는다.</p> <p>3. 다운로드한 HTML 페이지 내부의 오픈 그래프나 메타 태그 정보를 기반으로 페이지의 검색(공유) 정보를 가져오고 이를 바탕으로 검색 엔진에 저장한다.</p> <ul style="list-style-type: none"> • 누적 레이아웃 이동이 적다 :-> 누적 레이아웃이란 사용자에게 페이지를 보여준 이후에 뒤늦게 어떤 HTML 정보가 추가되거나 삭제되어 마치 화면이 덜컹거리는 것과 같은 부정적인 사용자 경험을 말한다. SSR은 요청이 완전히 완료된 이후에 완성된 페이지를 제공하기 때문에 비교적 자유롭다. • 사용자의 디바이스 성능에 비교적 자유롭다 :-> 자바스크립트 리소스 실행은 사용자의 디바이스에 서만 실행되므로 절대적으로 사용자 디바이스 성능에 의존적이다. 그러나 서버 사이드 렌더링을 수행하면 이러한 부담을 서버에 나눌 수 있으므로 사용자의 디바이스 성능으로부터 조금 더 자유로워질 수 있다. • 보안에 좀 더 안전하다-> JAM 스택을 채택한 프로젝트의 공통적인 문제점은 애플리케이션의 모든 활동이 브라우저에 노출된다는것인 반면, SSR은 인증 혹은 민감한 작업을 서버에서 수행하고 그 결과만 브라우저에 제공하기 때문에 보안 위협을 피할 수 있다는 장점이 있다. 			위한 리액트 API 살펴보기 (p. 293)
서버 사이드 렌더링의 단점은?	<ul style="list-style-type: none"> • 소스코드를 작성할 때 항상 서버를 고려해야 한다 :-> 가장 큰 문제인 브라우저 전역 객체인 window 또는 sessionStorage와 같이 브라우저에만 있는 전역객체를 접근할 수 없다, 외부에서 의존하고 있는 라이브러리도 고려해야한다. • 적절한 서버가 구축돼 있어야 한다 :-> 서버를 구축하는 것은 절대 쉬운 일이 아니다. • 서비스 지연에 따른 문제-> SSR은 서버에서 사용자에게 보여줄 페이지에 대한 렌더링 작업이 끝나기 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을



📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
	까지는 사용자에게 그 어떤 정보도 제공할 수 없다. 애플리케이션의 규모가 커지고 작업이 복잡해지고, 이에 따라 다양한 요청에 얽혀있어 병목 현상이 심해진다면 때로는 서버 사이드 렌더링이 더 나은 사용자 경험을 제공할 수도 있다.			위한 리액트 API 살펴보기 (p. 293)
<u>hydrate란?</u>	<ul style="list-style-type: none"> • 서버사이드 렌더링에서, 정적으로 생성된 HTML에 이벤트와 핸들러를 붙여 완전한 웹페이지 결과물을 만드는 것. 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>hydrate는 어떤 가정하에 실행 되는가?</u>	<ul style="list-style-type: none"> • 서버에서 제공해 준 HTML이 클라이언트의 결과물과 같을 것 이라는 가정하에 실행된다. 	@2024년 1월 8일	 지찬 주	3.2 사용자 정의 효과 고차 컴포넌트 중 무엇을 써야 할까? (p. 239) - 4.2 서버 사이드 렌더링을 위한 리액트 API 살펴보기 (p. 293)
<u>_app.tsx와 _document.tsx의 차이는 ?</u>	<ul style="list-style-type: none"> • _app.tsx 가 애플리케이션 페이지 전체를 초기화하는 곳이라면, _document.tsx는 애플리케이션의 HTML을 초기화하는 곳이다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 훑아보기 (p. 293) - 5.1 상



📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				태 관리는 왜 필요한가? (p. 348)
<u>다이나믹 라우팅?</u>	미리 정의된 URL 주소로만 라우팅 하는 것이 아니라 사용자가 접근한 경로 혹은 상황에 따라 동적인 라우팅 을 제공하고 싶을 때 사용할 수 있는 방식	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u>nextjs는 서버 라우팅과 클라이언트 라우팅이 혼재되어있는데 이에 따른 장점은?</u>	<ul style="list-style-type: none"> • SSR의 장점인 사용자가 빠르게 볼 수 있는 최초 페이지를 제공한다. • SPA의 장점인 자연스러운 라우팅을 제공한다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u><Link> <a/> 태그의 차이</u>	<ul style="list-style-type: none"> • <Link>는 클라이언트 라우팅/렌더링방식이다. • <a/>는 페이지를 만드는 데 필요한 모든 리소스를 처음부터 다 가져온다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u>Next.js에서 페이지 이동 시 지켜야하는 규칙</u>	<ul style="list-style-type: none"> • <a> 대신 <Link>를 사용한다. • window.location.push 대신 router.push를 사용한다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한


📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				가? (p. 348)
<u>getStaticPaths가 하는 기능</u>	<ul style="list-style-type: none"> • 정적으로 결정된 페이지를 보여 주고자 할 때 사용된다. • 접근 가능한 주소를 정의하는 함수 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u>getStaticProps가 하는 기능</u>	<ul style="list-style-type: none"> • 정적으로 결정된 페이지를 보여 주고자 할 때 사용된다. • 앞에서 정의한 페이지를 기준으로 해당 페이지로 요청이 왔을 때 제공할 props를 반환하는 함수 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u>getServerSideProps란?</u>	<ul style="list-style-type: none"> • 서버에서 실행되는 함수이며 페이지 진입 전에 이 함수를 실행한다. • 응답값에 따라 페이지의 루트 컴포넌트에 props를 반환할 수도, 다른 페이지로 리다이렉트 시킬 수도 있다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)
<u>getInitialProps란?</u>	<ul style="list-style-type: none"> • getStaticProps나 getServerSideProps가 나오기 전에 사용할 수 있었던 유일한 페이지 데이터 불러오기 수단 • 루트 함수에 정적 메서드로 추가된다. • props 객체를 반환하는 것이 아니라 객체를 반환한다. • 사용하기도 까다롭고 여러가지 주의할 점이 있으므로 가급적이면 Next.js의 특성상 사용이 제한돼 	@2024년 1월 9일	 지찬 주	4.3 Next.js 톺아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)

📖 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
	있는 페이지에서만 사용하는것이 좋다.			
<u>상태란?</u>	<ul style="list-style-type: none"> • 어떠한 의미를 지닌 값이며 애플리케이션의 시나리오에 따라 지속적으로 변경될 수 있는 값을 의미한다. • UI : 기본적으로 웹 애플리케이션에서 상태라 함은 상호 작용이 가능한 모든 요소의 현재 값을 의미한다. 다크/라이트 모드, 라디오를 비롯한 각종 input, 알림창의 노출 여부 등 많은 종류의 상태가 존재한다. • URL: 주소에 있는 상태값 • form : 로딩중인지, 현재 제출했는지, 접근이 불가능한지, 유효한지 등등, • 서버에서 가져온 값 : 클라이언트에서 서버로 요청을 통해 가져온 값도 상태다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 툼아보기 (p. 293) - 5.1 상태 관리 는 왜 필요한가? (p. 348)
<u>flux패턴 탄생 배경</u>	<ul style="list-style-type: none"> • 양방향 데이터 통신의 상태 변화 및 추적이 어려워 탄생하였다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 툼아보기 (p. 293) - 5.1 상태 관리 는 왜 필요한가? (p. 348)
<u>단방향 데이터 흐름의 장단점</u>	<ul style="list-style-type: none"> • 데이터의 흐름을 추적하기 쉽고 코드를 이해하기가 한결 수월해진다. • 사용자의 입력에 따라 데이터를 갱신하고 화면을 어떻게 업데이트 해야 하는지도 코드로 작성해야 하므로 코드의 양이 많아지고 개발자도 수고로워진다. 	@2024년 1월 9일	 지찬 주	4.3 Next.js 툼아보기 (p. 293) - 5.1 상태 관리 는 왜 필요한가? (p. 348)
<u>flux패턴의 용어</u>	<ul style="list-style-type: none"> • 액션 : 어떠한 작업을 처리할 액션과 그 액션 발생 시 함께 포함시킬 데이터를 의미한다. 액션 타입과 데이터를 각각 정의해 이를 디스패처로 보낸다. • 디스패처: 액션을 스토어에 보내는 역할을 한다. 콜백 함수 형태로 	@2024년 1월 9일	 지찬 주	4.3 Next.js 툼아보기 (p. 293) - 5.1 상태 관리

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
	<p>앞서 액션이 정의한 타입과 데이터를 모두 스토어에 보낸다.</p> <ul style="list-style-type: none"> • 스토어: 여기에서 실제 상태에 따른 값과 상태를 변경할 수 있는 메서드를 가지고 있다. 액션의 타입에 따라 어떻게 이를 변경할지가 정의돼 있다. • 뷰: 리액트의 컴포넌트에 해당하는 부분으로, 스토어에서 만들어진 데이터를 가져와 화면을 렌더링하는 역할을 한다. 또한 뷰에서도 사용자의 입력이나 행위에 따라 상태를 업데이트하고자 할 수 있을 것이다. 			<p>는 왜 필요한가? (p. 348)</p>
<p><u>서버에서만 실행되는 함수의 제약</u></p>	<ul style="list-style-type: none"> • window, document 객체에는 접근 불가 • domain 없이 fetch 요청 불가능 (본인의 호스트 유추 불가능) • 에러 발생 시 500과 같이 정의해둔 에러페이지로 리다이렉트 	@2024년 1월 9일	J JIN	<p>4.3 Next.js 튜아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)</p>
<p><u>전역 CSS 적용시키고 싶을 때 어느 파일에 하면 되는가?</u></p>	_app.tsx	@2024년 1월 9일	J JIN	<p>4.3 Next.js 튜아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)</p>
<p><u>SWR, 리액트 쿼리의 특화점?</u></p>	HTTP 요청에 특화되어 있다.	@2024년 1월 9일	J JIN	<p>4.3 Next.js 튜아보기 (p. 293) - 5.1 상태 관리는 왜 필요한가? (p. 348)</p>

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
상태관리api가 작동하는 방식을 간단히 요약하자면	<ul style="list-style-type: none"> • useState, useReducer가 가지고 있는 한계, 컴포넌트 내부에서만 사용할 수 있는 지역 상태라는 점을 극복하기 위해 외부 어딘가에 상태를둔다. 이는 컴포넌트의 최상단 내지는 상태가 필요한 부모가 될 수도 있고, 혹은 격리된 자바스크립트 스코프 어딘가일 수도 있다. • 이 외부의 상태 변경을 각자의 방식으로 감지해 컴포넌트의 렌더링을 일으킨다. 	@2024년 1월 10일	 지찬 주	5.2 리액트 혹은으로 시작하는 상태 관리 (p.349 - p. 399)
Recoil의 특징	<ul style="list-style-type: none"> • 리액트를 만든곳인 페이스북에서 만든 리액트를 위한 상태 관리 라이브러리이다. • 최소 상태 개념인 Atom을 처음 리액트 생태계에서 선보이기도 했다. • atom은 상태를 나타내는 Recoil의 최소 상태 단위다. • 아직 정식으로 출시한 라이브러리가 아니다. 따라서 실제 프로덕션에 사용하기에는 안정성이나 성능, 사용성 등을 보장할 수 없으며, 유의적 버전에 따라 부(minor) 버전이 변경돼도 호환성이 깨지는 변경 사항이 발생할 수도 있는 위험이 있다. 그러나 실제 프로젝트에서 안정적으로 서비스 하고 있다는 이야기도 들려오기 때문에 Recoil을 실제 프로덕션에 채택할 것인지는 개발자의 선택에 달려 있다. • Recoil을 사용하기 위해서는 RecoilRoot를 애플리케이션의 최상단에 선언해 줘야 한다. • Recoil의 상태값은 RecoilRoot로 생성된 Context의 스토어에 저장된다, • 스토어의 상태값에 접근할 수 있는 함수들이 있으며, 이 함수를 활용해 상태값에 접근하거나 상태값을 변경할 수 있다. • 값의 변경이 발생하면 이를 참조하고 있는 하위 컴포넌트에 모두 알린다. • selector를 사용해 한 개 이상의 atom값을 바탕으로 새로운 값을 조립할 수 있다. • selector를 필두로 강력한 비동기 작업을 지원하기 위한 API도 지원한다. 	@2024년 1월 10일	 지찬 주	5.2 리액트 혹은으로 시작하는 상태 관리 (p.349 - p. 399)

📖 문제	🗒️ 답변	📅 날짜	👤 Person	≡ 범위
	<ul style="list-style-type: none"> • Recoil에서도 자체적인 개발 도구를 지원한다. 			
<u>Jotai의 특징</u>	<ul style="list-style-type: none"> • Recoil의 atom 모델에 영감을 받아 만들어진 상태 관리 라이브러리다. • 상향식 접근법을 취하고있다, 작은 단위의 상태를 위로 전파할 수 있는 구조를 취하고 있다. • Context의 문제점인 불필요한 리렌더링이 일어난다는 문제를 해결하고자 설계돼 있으며, 추가적으로 개발자들이 메모이제이션이나 최적화를 거치지 않아도 리렌더링이 발생되지 않도록 설계돼 있다. • Recoil과 다르게 Jotai는 atom을 생성할 때 별도의 key를 넘겨주지 않아도 된다. • Recoil의 atom 개념을 도입하면 서도 API가 간결하다. Recoil의 atom에서는 각 상태값이 모두 별도의 키를 필요로 하기 때문에 이 키를 별도로 관리해야 하는데, Jotai는 이러한 부분을 추상화해 사용자가 키를 관리할 필요가 없다. Jotai가 별도의 문자열 키가 없이도 각 값들을 관리할 수 있는 것은 객체의 참조를 통해 값을 관리하기 때문이다. • selector없이 atom만으로 atom 값에서 또 다른 파생된 상태를 만들 수 있다. • 타입스크립트로 작성돼 있어 타입을 잘 지원하고 있다. • 리액트 18의 변경된 API를 원활하게 지원한다. • Recoil의 atom 형태의 상태 관리를 선호하지만, 아직 정식 버전이 출시되지 않아 사용이 망설여지는 많은 개발자들이 Jotai를 채택해 개발하고 있다. 	@2024년 1월 10일	 지찬 주	5.2 리액트 혹은으로 시작하는 상태 관리 (p.349 - p. 399)
<u>Zustand의 특징</u>	<ul style="list-style-type: none"> • 리덕스에 영감을 받아 만들어졌다. • 하나의 스토어를 중앙 집중형으로 활용해 이 스토어 내부에서 상태를 관리한다. • 특별히 많은 코드를 작성하지 않아도 빠르게 스토어를 만들고 사용할 수 있다는 큰 장점이 있다. • 라이브러리 크기가 타 API에 비해 월등히 작다. 즉, API가 복잡하지 않고 사용이 간단해 쉽게 접근할 수 있는 상태 관리 라이브러리 	@2024년 1월 10일	 지찬 주	5.2 리액트 혹은으로 시작하는 상태 관리 (p.349 - p. 399)

📖 문제	👍 답변	📅 날짜	👤 Person	≡ 범위
	<p>이다.</p> <ul style="list-style-type: none"> 타입스크립트 기반으로 작성돼 있기 때문에 별도로 @types를 설치하거나 임의로 작성된 d.ts에 대한 우려 없이 타입스크립트를 자연스럽게 쓸 수 있다. 리덕스와 마찬가지로 미들웨어를 지원한다. 			
<u>라이브러리를 선택하는데 효율적인 방법</u>	<ul style="list-style-type: none"> 각 라이브러리별로 특징을 잘 파악하고, 현재 애플리케이션의 상황과 철학에 맞는 상태 관리 라이브러리를 적절하게 선택해 사용하면 효율적인 애플리케이션을 만드는 데 도움이 될 것이다. 메인테이너가 많고 다운로드가 활발하며 이슈가 관리가 잘되고 있는 라이브러리를 선택하는 것이 좋다. 만약 이러한 대응이 원활하지 않은 라이브러리를 선택한다면 애플리케이션의 장기적인 유지보수 및 개선에 어려움이 있을 수 있다. 	@2024년 1월 10일	 지찬 주	5.2 리액트 혹은 시작하는 상태 관리 (p.349 - p. 399)
<u>Eslint란?</u>	<ul style="list-style-type: none"> 자바스크립트 생태계에서 가장 많이 사용되는 정적 코드 분석 도구 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>ESLint는 자바스크립트 코드를 어떻게 분석하는가?</u>	<ol style="list-style-type: none"> 자바스크립트 코드를 문자열로 읽는다. 자바스크립트 코드를 분석할 수 있는 파서(parser)로 코드를 구조화한다. 2번에서 구조화한 트리를 AST(Abstract Syntax Tree)라 하며, 이 구조화된 트리를 기준으로 각종 규칙과 대조한다. 규칙과 대조했을 때 이를 위반한 코드를 알리거나(report) 수정한다(fix) 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>plugins란</u>	<ul style="list-style-type: none"> • ESLint 규칙의 모음 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>트리셰이킹이란</u>	<ul style="list-style-type: none"> • 번들러가, 코드 어디에서도 사용하지 않는 코드(dead code, 이른바 죽은 코드)를 삭제해서 최종 번들 크기를 줄이는 과정을 의미한다. 나무의 나뭇잎을 털어낸다는 의미에서 유래했다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>백엔드와 프론트엔드의 테스트방법론 차이</u>	<ul style="list-style-type: none"> • 백엔드는 보통 화이트박스 테스트, 프론트엔드는 보통 블랙박스 테스트로 진행된다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>테스트 코드를 작성하기 전에 봐야 할 최우선 과제는?</u>	<ul style="list-style-type: none"> • 애플리케이션에서 가장 취약하거나 중요한 부분을 파악하는 것 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p)

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				- 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>eslint-plugin과 eslint-config란</u>	<ul style="list-style-type: none"> • eslint-plugin은 규칙을 모아놓은 패키지이며, eslint-config는 eslint-plugin을 한데 묶어서 완벽하게 한 세트로 제공하는 패키지라 할 수 있다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅 하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>eslint-config 라이브러리 중 대표적인것</u>	<ul style="list-style-type: none"> • eslint-config-airbnb 에어비엔비에서 만들어졌고 500여명의 수많은 개발자가 유지보수 하고 있다. • @titicaca/triple_config-kit 한국 커뮤니티에서 운영되는 eslint-config 중 유지보수가 활발한 편에 속한다, 자체적으로 정의한 규칙을 기반으로 운영되고 있다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅 하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>eslint사용할때 주의할점</u>	<ul style="list-style-type: none"> • Prettier와의 충돌이 있을 수 있다. : ESLint에서도 Prettier에서 처리하는 작업(들여쓰기, 줄바꿈, 따옴표, 최대 글자 수 등)을 처리할 수 있기 때문에 두가지 모두를 자바스크립트 코드에서 실행한다면 서로 충돌하는 규칙으로 인해 에러가 발생하고, 최악의 경우 ESLint, Prettier 모두 만족하지 못하는 코드가 만들어질 수도 있다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅 하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
				라이브러리 (528p)
ESLint의 특정 규칙을 임시로 제외시키고싶다면?	<ul style="list-style-type: none"> • eslint-disable- 주석을 사용한다. 	@2024년 1월 12일	👤 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
useEffect의 사용 시 위험한 케이스 3가지	<ul style="list-style-type: none"> • 의존성 배열이 없어도 괜찮다고 임의로 판단한 경우 : 실제로 면밀히 검토해서 괜찮은 경우라면 해당 변수는 컴포넌트의 상태와 별개로 동작한다는 것을 의미한다. 이 경우에는 해당 변수를 어디서 어떻게 선언할지 다시 고민해 봐야한다. 정말로 괜찮다 하더라도 이러한 작업이 반복되면 정말로 괜찮지 않은 코드에서도 동일하게 사용해 버그를 야기할 위험성이 있다. • 의존성 배열이 너무 긴 경우 : 의존성 배열이 너무 길다는 것은 useEffect 내부 함수가 너무 길다는 말과 동일하다. useEffect가 너무 길다면 useEffect를 쪼개서 의존성 배열의 가독성과 안정성을 확보해야 한다. • 마운트 시점에 한 번만 실행하고 싶은 경우 : 가장 흔히 볼 수 있는 경우로, 의도적으로 []로 모든 의존성을 제거해 컴포넌트가 마운트되는 시점에만 실행하고 싶은 경우다. 먼저 이러한 접근 방법은 과거 클래스형 컴포넌트에서 사용되던 생명주기 형태의 접근 방법으로, 함수형 컴포넌트의 패러다임과는 맞지 않을 가능성이 있다. 또한 [] 배열이 있다는 것은 컴포넌트의 상태값과 별개의 부수 효과가 되어 컴포넌트의 상태와 불일치가 일어날 수 있게 된다. 마지막으로, 상태와 관계없이 한번만 실행돼야 하는 것이 있다면 해당 컴포넌트에 존재 	@2024년 1월 12일	👤 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
	할 이유가 없다. 이 경우 적절한 위치로 옮기는 것이 좋다.			
<u>리액트 컴포넌트 테스트 순서</u>	<ol style="list-style-type: none"> 1. 컴포넌트를 렌더링한다. 2. 필요하다면 컴포넌트에서 특정 액션을 수행한다. 3. 컴포넌트 렌더링과 2번의 액션을 통해 기대하는 결과와 실제 결과를 비교한다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>데이터셋이란</u>	<ul style="list-style-type: none"> • HTML의 특정 요소와 관련된 임의의 정보를 추가할 수 있는 HTML 속성이며, HTML의 특정 요소에 data-로 시작하는 속성은 무엇이든 사용할 수 있다. 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>그 외에 해볼 만한 여러 가지 테스트는?</u>	<ul style="list-style-type: none"> • 유닛 테스트: 각각의 코드나 컴포넌트가 독립적으로 분리된 환경에서 의도된 대로 정확히 작동하는지 검증하는 테스트 • 통합 테스트: 유닛 테스트를 통과한 여러 컴포넌트가 묶여서 하나의 기능으로 정상적으로 작동하는지 확인하는 테스트 • 엔드 투 엔드: 흔히 E2E 테스트라 하며, 실제 사용자처럼 작동하는 로봇을 활용해 애플리케이션의 전체적인 기능을 확인하는 테스트 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기 (461p) - 8.2 리액트 팀이 권장하는 리액트 테스트 라이브러리 (528p)
<u>테스트할 수 있는 방법은 여러가지가 있지만 테스트가 이뤄야 할 목표는?</u>	<ul style="list-style-type: none"> • 애플리케이션이 비즈니스 요구사항을 충족하는지 확인하는 것 	@2024년 1월 12일	 지찬 주	7.6 Next.js 환경 디버깅하기

📖 문제	✔️ 답변	📅 날짜	👤 Person	≡ 범위
				기 (461p) - 8.2 리액트 팀이 권 장하는 리액트 테스트 라이브 러리 (528p)
<u>MSW 는 언제 사용할 수 있나요?</u>	http 요청을 모킹하여 테스트 하고 싶을 때	@2024년 1월 12일	 JIN	7.6 Next.js 환경 디 버깅 하 기 (461p) - 8.2 리액트 팀이 권 장하는 리액트 테스트 라이브 러리 (528p)
<u>직접 작성하지 않고 유용 한 액션과 깃허브 앱 가 저다 쓸 수 있는곳</u>	• MarketPlaces	@2024년 1월 13일	 지찬 주	9. 모던 리액트 개발 도 구로 개 발 및 배포 환 경 구축 하기 (p. 529) - (p.654)
<u>패키지를 구분하는 것에 의문을 제기하는 목소리</u>	<ul style="list-style-type: none"> • 번들러의 존재: devDependencies로 설치한 것이 든, dependencies로 설치한 것이 든 모두 node_modules에 동일하 게 설치한다. 둘의 차이는 최종 결 과물에는 전혀 영향을 미치지 않는 다. • 복잡해진 개발 파이프라인 • devDependencies와 dependencies의 경계가 모호해 지고있다. 	@2024년 1월 13일	 지찬 주	9. 모던 리액트 개발 도 구로 개 발 및 배포 환 경 구축 하기 (p. 529) - (p.654)
<u>의존성 이슈를 방지하는 가장 좋은 방법</u>	• 의존성을 최소한으로 유지하는 것. 바깥에 노출되는 면적이 클수 록 위험에 노출되는 확률이 커지는	@2024년 1월 13일	 지찬 주	9. 모던 리액트 개발 도

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
	것과 마찬가지로 의존성, 즉 dependencies와 node_modules의 크기가 커질수록 위협에 노출될 확률 또한 높아진다. 가능한 한 내재화할 수 있는 모듈은 내재화하고, 의존성을 최소한으로 유지하는 것이 좋다.			구로 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>dependabot은 이슈를 찾는 용도로만 사용하고 절대로 완벽하게 수정해 준다고 맹신해서는 안되는 이유를 이 책에서는 어떻게 설명하였는가</u>	<ul style="list-style-type: none"> • 예를들면 유의적 버전에 따라 주 버전을 올리는 것은, 앞서 버전에 관해 알아본 것처럼 실제 라이브러리를 사용하는 데 많은 변경이 있을 수 있기 때문에 무작정 머지해서는 안 된다. 	@2024년 1월 13일	 지찬 주	9. 모던 리액트 개발 도구로 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>도커란 도커 스스로를 어떤 서비스라고 정의하는가</u>	<ul style="list-style-type: none"> • 도커는 개발자가 모던 애플리케이션을 구축, 공유, 실행하는 것을 도와줄 수 있도록 설계된 '플랫폼'이다. 도커는 지루한 설정 과정을 대신해 주므로 코드를 작성하는 일에만 집중할 수 있다. 	@2024년 1월 13일	 지찬 주	9. 모던 리액트 개발 도구로 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>리액트 애플리케이션 빠르게 배포하는 방법</u>	Netlify, Vercel, DigitalOcean	@2024년 1월 13일	 JIN	9. 모던 리액트 개발 도구로 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>도커 장점</u>	어디서든 실행 가능한 상태로 애플리케이션을 준비해둔다면 도커 이미지를 실행할 수 있는 최소한의 환경이 갖춰진 상태라면 어디서든 배포가 가능하다.	@2024년 1월 13일	 JIN	9. 모던 리액트 개발 도구로 개발 및 배포 환경 구축하기 (p. 529)

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
) - (p.654)
<u>유의적 버전 정의에 대해서 말해주세요</u>	<ul style="list-style-type: none"> • 주 : 기존 버전과 호환되지 않게 API가 바뀌면 올린다. • 부 : 기존 버전과 호환되면서 새로운 기능을 추가할 때 올린다. • 수 : 기존 버전과 호환되면서 버그를 수정한 경우 올린다. 	@2024년 1월 13일	J JIN	9. 모던 리액트 개발 도구 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>주 버전이 0인 경우는 언제 쓰는가?</u>	초기 개발을 위해 쓴다. 안정적이지 않다.	@2024년 1월 13일	J JIN	9. 모던 리액트 개발 도구 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>package.json에서 dependencies, devDependencies, peerDependencies의 차이점</u>	<ul style="list-style-type: none"> • dependencies : 프로젝트를 실행을 위해 꼭 필요한 패키지 • devDependencies : 실행에는 필요하지 않지만 개발에는 필요한 패키지 • peerDependencies : 서비스보다는 라이브러리, 패키지에서 주로 쓰이며 호환을 목적으로 필요한 패키지 	@2024년 1월 13일	J JIN	9. 모던 리액트 개발 도구 개발 및 배포 환경 구축하기 (p. 529) - (p.654)
<u>리액트 18 버전에서, 자동 배치란?</u>	리액트가 여러 상태 업데이트를 하나의 리렌더링으로 묶어서 성능을 향상시키는 방법을 의미한다. 예를 들어, 버튼 클릭 한 번에 두 개 이상의 state를 동시에 업데이트 한다고 가정해 보자. 자동 배치에서는 이를 하나의 리렌더링으로 묶어서 수행할 수 있다.	@2024년 1월 13일	지찬 주	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>엄격모드에서 두번씩 실행되는 이유</u>	함수형 프로그래밍의 원칙에 따라 리액트의 모든 컴포넌트는 항상 순	@2024년 1월 15일	지찬 주	10.1 리액트

📄 문제	✔️ 답변	📅 날짜	👤 Person	≡ 범위
	수하다고 가정하기 때문에 엄격모드에서는 이것이 실제로 지켜지고 있는지, 즉 항상 순수한 결과물을 내고 있는지 개발자에게 확인시켜 주기 위해 두 번 실행한다.			17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>테어링이란</u>	리액트에서는 하나의 state 값이 있음에도 서로 다른 값(보통 state 나 props의 이전과 이후)을 기준으로 렌더링 되는 현상을 말한다.	@2024년 1월 15일	 지찬 주	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>리액트 18 버전에서, 엄격모드 뜻과 종류</u>	- 리액트 애플리케이션에서 발생할 수도 있는 잠재적인 버그를 찾는 데 도움이 되는 컴포넌트 더 이상 안전하지 않은 특정 생명 주기를 사용하는 컴포넌트에 대한 경고 : UNSAFE_가 붙은 생명주기 메서드를 사용하면 오류 발생 문자열 ref 사용 금지 findDOMNode에 대한 경고 출력 구 Context API 사용 시 발생하는 경고 예상치 못한 부작용(side-effects) 검사	@2024년 1월 15일	 지찬 주	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>리액트 팀에서 버전업을 할 때 사용자들에게 추천하는 개발 방향</u>	점진적 채택, 점진적 버전업이 가능하다.	@2024년 1월 15일	 J JIN	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)

📄 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
<u>이벤트 단계</u>	<ul style="list-style-type: none"> • 캡처 : 이벤트 핸들러가 트리 최상단 요소에서 시작해서 실제 이벤트가 발생한 타겟 요소까지 내려가는 것 • 타겟 : 이벤트 핸들러가 타겟 노드에 도달하는 단계, 이 때 이벤트가 호출된다. • 버블링 : 이벤트가 발생한 요소에서 부터 시작해서 최상위 요소까지 올라간다. 	@2024년 1월 15일	 JIN	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>이벤트 위임에 대한 리액트 16, 17 의 차이</u>	<ul style="list-style-type: none"> • 16버전에서는 이벤트 위임이 document에서 수행 • 17버전에서는 리액트 컴포넌트 최상단 트리 ⇒ 변경한 이유 : 점진적인 업그레이드를 지원하기 위해, 다른 버전 코드와 혼재되어있을 때 혼란을 줄이기 위해서	@2024년 1월 15일	 JIN	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>컴포넌트의 undefined 반환에 대한 처리 차이 16, 17, 18</u>	<ul style="list-style-type: none"> • 16 : 오류가 발생한다. forwardRed나 memo에서 undefined를 반환하는 경우는 에러 발생하지 않았다. • 17 : 모든 경우에 에러 발생 • 18 : 에러가 나지 않는다. null을 반환한 것과 같다. 	@2024년 1월 15일	 JIN	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657) - (p. 713)
<u>18버전의 가장 큰 변경점</u>	<ul style="list-style-type: none"> • 동시성 지원 • 과거 리액트 모든 렌더링을 동기적으로 작동하여 느린 렌더링 작업이 있을 경우 서비스 전체에 영향을 미쳤지만 동시성을 지원하는 기능을 사용하면 느린 렌더링 과정에서 로딩 화면을 보여주거나 진행 중인 렌더링을 버리고 새로운 상태 값으로 재렌더링 하는 등 성능 향상 뿐만 아니라 사용자에게 좀 더 자연스러운 서비스 경험 	@2024년 1월 15일	 JIN	10.1 리액트 17 버전 살펴보기 - 10.2 리액트 18 버전 살펴보기 (p. 657)

📖 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
) - (p. 713)
<u>nextjs에서 페이지의 기본적인 레이아웃을 구성하는 요소</u>	layout.js	@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)
<u>클라이언트 컴포넌트를 명시적으로 선언하는 방법</u>	• 'use client'	@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)
<u>• 데이터의 유효한 시간을 정해두고 이 시간이 지나면 다시 데이터를 불러와서 페이지를 렌더링 하는것이 가능한 변수이름</u>	revalidate	@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)
<u>서버컴포넌트의 작동방식의 특별한점</u>	<ul style="list-style-type: none"> • 서버에서 클라이언트로 정보를 보낼 때 스트리밍 형태로 보냄으로써 클라이언트가 줄 단위로 JSON을 읽고 컴포넌트를 렌더링할 수 있어 브라우저에서는 되도록 빨리 사용자에게 결과물을 보여줄 수 있다 • 컴포넌트들이 하나의 번들러 작업에포함돼 있지 않고 각 컴포넌트 별로 번들링이 별개로 돼 있어 필요에 따라 컴포넌트를 지연해서 받거나 따로 받는 등의 작업이 가능하다. • SSR과는 다르게 결과물이 HTML이 아닌 JSON형태로 보내진 것 또한 주목해 볼 만하다. 클라이언트의 최종 목표는 리액트 컴포넌트 트리를 서버 컴포넌트와 클라이언트 컴포넌트의 두 가지로 조화롭게 구성하는 것으로, 이는 단순히 HTML을 그리는 작업 이상의 일을 필요로 한다. 따라서 HTML 	@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)

📖 문제	🗳️ 답변	📅 날짜	👤 Person	≡ 범위
	대신 단순한 리액트 컴포넌트 구조를 JSON으로 받아서 리액트 컴포넌트 트리의 구성을 최대한 빠르게 할 수 있도록 도와준다.			
<ul style="list-style-type: none"> • <u>서버컴포넌트에서 prop를 넘길때 반드시 직렬화가능한 데이터를 넘겨야 한다</u> 		@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)
<u>서버 컴포넌트의 제약사항, 클라이언트 컴포넌트의 제약사항, 공용 컴포넌트의 제약사항 과 차이점</u>	<ul style="list-style-type: none"> • 서버컴포넌트 : <ol style="list-style-type: none"> 1. 요청이 오면 그 순간 서버에서 딱 한 번 실행될 뿐이므로 상태를 가질 수 없다. 따라서 리액트 훅을 사용할 수 없다. 2. 사용자 정의훅 또한 사용불가 다만 effect나 state에 의존하지 않고 서버에서 제공할 수 있는 기능만 사용하는 훅이라면 충분히 사용 가능하다. 3. 서버에서만 실행되기 때문에 DOM API나 window, document 등에 접근할 수 없다. 4. 데이터베이스, 내부 서비스, 파일 시스템 등 서버에만 있는 데이터를 async/await으로 접근할 수 있다. • 클라이언트 컴포넌트 <ol style="list-style-type: none"> 1. 브라우저 환경에서만 실행되므로 서버컴포넌트를 불러오거나 서버 전용 훅이나 유틸리티를 불러올 수 없다. 2. 서버 컴포넌트가 클라이언트 컴포넌트를 렌더링하는데, 그 클라이언트 컴포넌트가 자식으로 서버 컴포넌트를 갖는 구조는 가능하다, 그 이유는 클라이언트 입장에서 봤을 때 서버 컴포넌트는 이미 서버에서 만들어진 트리를 가지고 있을 것이고, 클라이언트 컴포넌트는 이미 서버에서 만들어진 그 트리를 삽입해서 보여주지만 하기 때문이다. • 공용 컴포넌트 <ol style="list-style-type: none"> 1. 서버컴포넌트와 클라이언트 컴포넌트의 모든 제약을 받는 컴포넌트 	@2024년 1월 16일	👤 지찬 주	11.1 app 디렉터리의 등장 - 11.8 정리 (p. 715) - (p.774)

📄 문제	✅ 답변	📅 날짜	👤 Person	≡ 범위
<u>구글은 사용자 경험을 크게 4가지로 분류해 정의하는데, 이를 RAIL이라고 한다. RAIL이란?</u>	<ul style="list-style-type: none"> • Response: 사용자의 입력에 대한 반응 속도. 50ms 미만으로 이벤트를 처리할 것 • Animation: 애니메이션의 각 프레임이 10ms 이하로 생성할 것 • Idle: 유휴 시간을 극대화해 페이지가 50ms 이내에 사용자 입력에 응답하도록 할 것 • Load: 5초 이내에 콘텐츠를 전달하고 인터랙션을 준비할 것 	@2024년 1월 18일	 지찬 주	12.1 웹사이트와 성능 - 12.6 정리 (p. 775) - (p.812)
<u>최초 콘텐츠폴 페인트 (First Contentfull Paint, FCP)란?</u>	<ul style="list-style-type: none"> • 페이지가 로드되기 시작한 시점부터 페이지 콘텐츠의 일부가 화면에 렌더링될 때까지의 시간을 측정한다. 조금 더 쉽게 설명하자면 웹사이트에 접속한 순간부터 페이지에 뭐라도 뜨기 시작한 시점까지의 시간을 의미한다. 	@2024년 1월 18일	 지찬 주	13. 웹 페이지의 성능을 측정하는 다양한 방법 (p. 813 - p. 873)
<u>크로스사이트 스크립팅이란</u>	<ul style="list-style-type: none"> • 웹사이트 개발자가 아닌 제3자가 웹사이트에 악성 스크립트를 삽입해 실행할 수 있는 취약점을 의미한다. 	@2024년 1월 19일	 지찬 주	14. 웹사이트 보안을 위한 리액트와 웹페이지 보안 이슈 (p. 875) - 15. 마치며 (p. 916)
<u>리액트에서 XSS 문제를 피하는 방법</u>	<ul style="list-style-type: none"> • 제 3자가 삽입할 수 있는 HTML을 안전한 HTML 코드로 한 번 치환하는 것, 이러한 과정을 새니타이즈, 또는 이스케이프라고한다. 	@2024년 1월 19일	 지찬 주	14. 웹사이트 보안을 위한 리액트와 웹페이지 보안 이슈 (p. 875) - 15. 마치며 (p. 916)
<u>리액트 프로젝트를 시작할때 고려해야할 사항</u>	<ul style="list-style-type: none"> • 리액트 버전을 최소 16.8.6에서 최대 17.0.2로 올려두자 : 대규모 업데이트가 벌어진 버전 • 인터넷 익스플로러 11 지원을 목표한다면 각별히 더 주의를 기한 	@2024년 1월 19일	 지찬 주	14. 웹사이트 보안을 위한 리액트와

📖 문제	✔️ 답변	📅 날짜	👤 Person	≡ 범위
	<p>다.</p> <ul style="list-style-type: none"> • 서버 사이드 렌더링 애플리케이션을 우선적으로 고려한다 : 사용자 기기가 성능이 천차만별이기때문 • 상태 관리 라이브러리는 꼭 필요할 때만 사용한다. : • 리액트 의존성 라이브러리 설치를 조심한다. : 버전확인 필수 			<p>웹페이지 보안 이슈 (p. 875) - 15. 마치며 (p. 916)</p>