질문	답변
동기 처리와 비동기 처리의 차이는 무엇인가요?:	동기 처리는 작업을 **순서대로** 진행하는 방식으로, 한 작업이 **끝나야** 다음 작업이 **시작**됩니다. 예를 들어, 파일을 읽는 동기 처리의 경우, 파일 읽기 작업이 완료될 때까지 프로그램은 다른 작업을 수행하지 못하고 대기합니다. 이 방식은 코드의 실행 순서가 명확하므로 이해하고 디버그하기 쉽습니다. 하지만, 긴 시간이 걸리는 작업이 있을 경우 프로그램이 **블로킹**되어 사용자 경험이 좋지 않을 수 있습니다. 반면, 비동기 처리는 한 작업이 끝나지 **않아도 다음 작업을 시작**하는 방식입니다. 예를 들어, 네트워크 요청과 같은 비동기 작업의 경우, 요청을 보낸 후 응답이 올 때까지 **다른 작업을 계속 수행**할 수 있습니다. 이 방식은 **블로킹 없이** 효율적으로 작업을 수행할 수 있지만, 코드의 **실행 순서가 복잡**해져서 이해하고 디버그하기 어려울 수 있습니다.
비동기 처리를 사용하는 이유는 무엇인가요?	비동기 처리는 주로 **I/O 작업**(파일 읽기/쓰기, 네트워크 요청 등)에서 사용됩니다. I/O 작업은 CPU 작업에 비해 상대적으로 **많은 시간**이 필요하므로, 이런 작업을 동기적으로 처리하면 프로그램이 **블로킹**되어 **사용자 경험**이 좋지 않을 수 있습니다. 따라서, 이런 작업을 비동기적으로 처리하여 프로그램의 응답성을 향상시키고, 시스템 **자원을 효율**적으로 활용할 수 있습니다.
Promise와 async/await의 역할과 차이점은 무엇인가요?:	Promise는 비동기 작업의 **최종 완료 또는 실패**를 나타내는 객체입니다. Promise는 비동기 작업이 완료되면 **결과** 값을 가지며, 이 결과 값은 **.then() 메서드**를 통해 접근할 수 있습니다. 또한, **.catch()** 메서드를 사용하여 비동기 작업 중 발생한 **에러를 처리**할 수 있습니다. async/await는 Promise를 더 간결하고 **동기적인 방식**으로 작성할 수 있게 해주는 문법입니다. async 함수는 항상 **Promise를 반환**하며, await 키워드는 Promise가 처리될 때까지 async 함수의 실행을 **일시 중지**합니다. async/await는 콜백 지옥을 피하고, 동기적인 방식으로 비동기 코드를 작성할 수 있게 해주므로 코드가 더 간결하고 이해하기 쉬워집니다.
동기 처리와 비동기 처리의 예시를 들어주세요:	동기 처리의 예로는 배열의 forEach() 메서드가 있습니다. forEach() 메서드는 배열의 모든 **요소를 순서대로 처리**하며, 한 요소를 처리한 후에 다음 요소를 처리합니다. 비동기 처리의 예로는 fetch() 함수가 있습니다. fetch() 함수는 네트워크 요청을 보내고, **요청이 완료되면 Promise를 반환**합니다. 이 Promise는 요청의 **응답**을 나타내며, .then() 메서드를 사용하여 응답을 처리할 수 있습니다.
블로킹(Blocking)과 논블로킹(Non-blocking)의 차이는 무엇인가요?	블로킹은 특정 작업이 완료될 때까지 프로그램의 **실행을 중지**하는 현상을 말합니다. 예를 들어, 동기적으로 파일을 읽는 작업은 파일 읽기가 완료될 때까지 프로그램이 다른 작업을 수행하지 못하도록 블로킹합니다. 반면, 논블로킹은 특정 작업이 **완료되지 않아도** 프로그램의 **실행을 계속 진행**하는 것을 말합니다. 예를 들어, 비동기적으로 네트워크 요청을 보내는 작업은 요청이 완료되지 않아도 프로그램이 다른 작업을 계속 수행할 수 있도록 논블로킹합니다.

웹 성능이 무엇이고 왜 중요한가요?	웹 성능은 사용자가 원하는 콘텐츠나 서비스를 **신속**하게 전달받을 수 있는 시스템의 성능을 나타냅니다. 웹 **로딩 시간**이 빠르면 사용자의 **이탈률**이 낮아지며, 이는 **매출**과 직결된 중요한 문제가 됩니다. 실제로 로딩이 **2**초 내로 완료되면 사용자의 구매 확률이 가장 높아집니다. 반면 로딩이 느리면 사용자 불만과 부정적 이미지, 선입견이 발생하며, **3초** 안에 웹 사이트가 로딩되지 않으면 이탈률이 급격히 증가할 수 있습니다.
프론트엔드 관점에서 웹 최적화의 지표에 대해 설명해주세요	프론트엔드 **빠르고 보기 쉽게 콘텐츠**를 **전달**하는것. 감각적인 디자인으로 웹 사이트 **유입 증가**와 콘텐츠를 **보기 편하게** 전달하는 역할. 백엔드: 서버처리량이나 네트워크 스위치 처리량, 처리속도에 문제가 없는지 등을 확인해야 한다. 전달환경- 네트워크: 대역폭과 지연시간을 측정한다.
프론트엔드가 웹 최적화 하는 방법에 대해 알려주세요	**HTTP 요청 수 줄이기** 웹 페이지에서 **요청**하는 콘텐츠 수를 줄이는 것이 **로딩 시간 단축**에 중요합니다. **스크립트 파일 병합:** 여러 **모듈화된 스크립트 파일**을 하나로 합쳐 HTTP 요청을 줄임. 파일 크기를 적절히 유지해야 합니다. **인라인 이미지:** 이미지의 해시 정보를 이용하여 인라인 이미지로 삽입. 인터넷 프록시 서버나 브라우저에 캐싱하기 어려울 수 있으 **CSS 스프라이트:** 여러 이미지를 하나의 파일로 결합하고 필요한 위치를 픽셀 좌표 정보로 사용. **콘텐츠 파일 크기 줄이기:** 큰 파일은 웹 성능에 부정적인 영향을 미칩니다. **스크립트 파일 압축:** 서버에서 스크립트 형태 콘텐츠를 압축하여 클라이언트에게 작은 크기로 전송. 클라이언트 및 서버가 지원하는 **스크립트 파일 최소화:** 주석, 공백, 개행문자 등 스크립트 파일에서 로직과 무관한 부분 제거. 개발 서버와 운영 서버 분리하여 작업. **이미지 파일 압축:** 이미지 파일의 메타데이터를 제거하여 크기 감소. 브라우저가 선호하는 이미지 파일 사용: WebP 및 JPEG XR 등의 형식 활용. **캐시 최적화하기:** 캐시를 최적화하여 웹 성능을 향상시킵니다. **인터넷 캐시 사용:** 프록시 서버를 활용하여 자주 요청되는 콘텐츠를 캐시하고 빠른 응답을 제공. **브라우저 캐시 사용:** 클라이언트 측에서 콘텐츠를 저장하여 인터넷 요청을 줄임. Cache-Control 헤더를 활용하여 캐시 정책 설정. **CDN 사용하기:**콘텐츠 전송 네트워크 (CDN)를 사용하여 웹 콘텐츠를 더 빠르게 전달합니다. CDN은 대용량 인터넷 캐시 영역에 콘텐츠를 저장하고 사용자에게 빠르게 제공하는 네트워크 방식으로, 사용자의 ISP 데이터 센터
프론트엔드에서 웹 최적화를 위해 주로 어떤 도구나 기술을 사용하나요?	예를 들어, 웹팩(Webpack)과 같은 번들링 도구로 스크립트 파일을 최적화하고, 이미지 최적화 도구를 사용하여 이미지를 압축할 수 있습니다. 또한 CDN(Content Delivery Network)를 활용하여 콘텐츠를 전송하고, 웹 성능 분석 도구를 사용하여 병목 현상을 찾아 최적화할 수 있습니다.