질문 및 개념	답변	중요도	관련 개념
HTTP와 HTTPS에 대해 설명해주세요.	HTTP는 웹상에서 데이터를 주고 받기 위한 프로토콜, 모바일이나 게임환경에서도 통신을 할 때 사용됩니다. **Stateless:** 클라이언트에서 서버로의 각 요청은 이전 요청에 대한 지식 없이 독립적인 트랜잭션으로 처리됩니다. 이는 서버가 요청 간에 클라이언트에 대한 상태 정보를 유지하지 않음을 의미합니다. **Connectionless:** HTTP는 클라이언트와 서버 간의 지속적인 연결이 필요하지 않습니다. 대신 각 요청에 대해 새 연결이 설정되고 응답이 반환된 후 닫힙니다. **텍스트 기반:** HTTP 메시지는 일반 텍스트로 전송되어 사람이 읽을 수 있고 디버그하기 쉽습니다. **확장 가능:** HTTP는 새로운 방법과 헤더의 정의를 허용하므로 프로토콜을 확장하여 새로운 사용 사례와 애플리케이션을 지원할 수 있습니다. HTTPS는 HTTP 프로토콜의 보안 버전인 Hypertext Transfer Protocol Secure의 약자입니다. 웹 서버와 웹 브라우저 간에 보안 연결을 설정하는 데 사용되어 이들 간에 교환되는 모든 데이터가 암호화되고 도청 및 변조로부터 보호됩니다. HTTPS는 SSL/TLS(Secure Sockets Layer/Transport Layer Security) 프로토콜을 사용하여 서버와 클라이언트 간에 교환되는 데이터를 암호화합니다. 이 암호화는 권한이 없는 사람이 데이터를 가로채거나 읽을 수 없도록 합니다. HTTPS 연결을 설정하려면 웹 서버에 SSL/TLS 인증서가 설치되어 있어야 합니다. 이 인증서는 인증 기관(CA)이라고 하는 신뢰할 수 있는 타사 조직에서 발급합니다. 웹 브라우저가 HTTPS를 사용하여 서버에 연결할 때 SSL/TLS 인증서가 유효하고 신뢰할 수 있는 CA에서 발급되었는지 확인합니다. **인증:** HTTPS는 클라이언트가 사기꾼이 아닌 의도된 서버와 통신하고 있는지 확인하기 위해 서버의 신원을 확인합니다. **무결성:** HTTPS는 서버와 클라이언트 간에 전송되는 데이터가 전송 중에 변조되지 않았음을 보장합니다. **기밀성:** HTTPS는 서버와 클라이언트 간에 교환되는 데이터를 기밀로 유지하고 다른 사람이 읽을 수 없도록 합니다.	상	
HTTP와 HTTPS의 차이점은 무엇인가요	HTTP는 인터넷 상에서 클라이언트와 서버가 자원을 주고 받을 때 쓰는 통신 규약입니다. HTTP는 텍스트 교환이므로, 누군가 네트워크에서 신호를 가로채면 내용이 노출되는 보안 이슈가 존재합니다. 이런 보안 문제를 해결해주는 프로토콜이 HTTPS입니다	상	
	HTTP는 평문 데이터를 전송하는 프로토콜이기 때문에, HTTP로 비밀번호나 주민번호 등을 주고 받으면 제3자에 의해 조회될 수 있습니다. 이러한 문제를 해결하기 위해 HTTP에 암호화가 추가된 프로토콜이 HTTPS입니다. HTTPS에는 대칭키 암호화와 비대칭키 암호화가 모두 사용됩니다. 비대칭키 암/복호화는 비용이 매우 크기 때문에 서버와 클라이언트가 주고받는 모든 메세지를 비대칭키로 암호화하면 오버헤드가 발생할 수 있습니다. 그래서 서버와 클라이언트가 최초 1회로 서로 대칭키를 공유하기 위한 과정에서 비대칭키 암호화를 사용하고, 이후에 메세지를 주고 받을 때에는 대칭키 암호화를 사용합니다. 이러한 과정을 정리하면 다음과 같습니다. - 클라이언트가 서버로 최초 연결 시도를 함 - 서버는 공개키를 넘겨줌 - 클라이언트는 인증서의 유효성을 검사하고 세션키를 발급함 - 클라이언트는 서버의 공개키로 세션키를 암호화하여 서버로 전송함 - 서버는 암호화된 세션키를 개인키로 복호화하여 세션키를 얻음 - 클라이언트와 서버는 동일한 세션키를 공유하므로 데이터를 전달할 때 세션키로 암호화/복호화를 진행함		
	HTTP 동작 순서 : TCP → HTTP HTTPS 동작 순서 : TCP → SSL → HTTP SSL(Secure Socket Layer)을 쓰냐 안쓰냐의 차이다. SSL 프로토콜은 정보를 암호화시키고 이때 공개키와 개인키 두가지를 이용한다. HTTPS는 인터넷 상에서 정보를 암호화하기 위해 SSL 프로토콜을 이용해 데이터를 전송하고 있다는 것을 말한다. 즉, 문서 전송시 암호화 처리 유무에 따라 HTTP와 HTTPS로 나누어지는 것 모든 사이트가 HTTPS로 하지 않는 이유는, 암호화 과정으로 인한 속도 저하가 발생하기 때문이다.		

HTTP의 문제점에는 무엇이 있을까요?	HTTP는 크게 세 가지의 보안 취약점을 가지고 있습니다. 1. 도청이 가능하다 - 평문으로 통신하기 때문에 도청이 가능하다 - 이를 해결하기 위해서 통신자체를암호화 (HTTPS)하거나 데이터를 암호화 하는 방법등이 있다 - 데이터를 암호화 하는 경우 수신측에서는 보호화 과정이 필요하다 2. 위장이 가능하다 - 통신 상대를 확인하지 않기 깨문에 위장된 상대와 통신할 수 있다 - HTTPS는 CA 인증서를 통해 인증된 상대와 통신이 가능하다 3. 변조가 가능하다 - 완전성을 보장하지 않기 때문에 변조가 가능하다 - HTTPS는 메세지 인증 코드(MAC), 전자 서명등을 통해 변조를 방지 한다	중	
HTTP Method들에 대해 설명해주세요.	GET: GET 방식은 서버에서 리소스를 가져오는 데 사용됩니다. 요청된 리소스는 요청에 포함된 URI로 식별됩니다. GET 요청은 데이터만 검색해야 하며 서버에 부작용이 있어서는 안 됩니다. POST: POST 방식은 리소스를 생성하거나 업데이트하기 위해 서버에 데이터를 제출하는 데 사용됩니다. PUT: PUT 방식은 서버의 기존 리소스를 업데이트하는 데 사용됩니다. DELETE: DELETE 메소드는 서버에서 리소스를 삭제하는 데 사용됩니다. PATCH: PATCH 방식은 서버에 있는 기존 리소스의 일부를 업데이트하는 데 사용됩니다. OPTIONS: OPTIONS 메소드는 URI로 식별되는 리소스에 사용 가능한 통신 옵션에 대한 정보를 검색하는 데 사용됩니다. 여기에는 지원되는 메서드, 헤더 및 콘텐츠 유형과 같은 정보가 포함될 수 있습니다.	중	
GET과 POST의 차이는?	둘다 HTTP 프로토콜을 이용해 서버에 무언가 요청할 때 사용하는 방식입니다 GET 방식은, URL을 통해 모든 파라미터를 전달하기 때문에 주소창에 전달 값이 노출됨. URL 길이가 제한이 있기 때문에 전송 데이터 양이 한정되어 있고, 형식에 맞지 않으면 인코딩해서 전달해야 합니다 POST 방식은 HTTP BODY에 데이터를 포함해서 전달함. 웹 브라우저 사용자의 눈에는 직접적으로 파라미터가 노출되지 않고 길이 제한이 없습니다 보통 GET은 가져올 때, POST는 수행하는 역할에 활용합니다 GET은 SELECT 성향이 있어서 서버에서 어떤 데이터를 가져와서 보여주는 용도로 활용합니다. POST는 서버의 값이나 상태를 바꾸기 위해 활용합니다.	중	

캡슐화란 무엇인가요 (객체지향 관점)	캡슐화(Encapsulation)란 낮은 결합도를 유지할 수 있도록 설계하는 것입니다 쉽게 말하면, **한 곳에서 변화가 일어나도 다른 곳에 미치는 영향을 최소화 시키는 것**을 말합니다 (객체가 내부적으로 기능을 **어떻게 구현하는지 감추는 것**) 결합도가 낮도록 만들어야 하는 이유가 무엇일까? 결합도(coupling)란, 어떤 기능을 실행할 때 다른 클래스나 모듈에 얼마나 의존적인가를 나타내는 말입니다. 즉, 독립적으로 만들어진 객체들 간의 의존도가 최대한 낮게 만드는 것이 중요합니다. 객체들 간의 의존도가 높아지면 굳이 객체 지향으로 설계하는 의미가 없어집니다. 우리는 소프트웨어 공학에서 객체 안의 모듈 간의 요소가 밀접한 관련이 있는 것으로 구성하여 응집도를 높이고 결합도를 줄여야 요구사항 변경에 대처하는 좋은 설계 방법이라고 배웁니다. 이것이 바로 `캡슐화`와 크게 연관된 부분이라고 볼 수 있습니다. 그렇다면, 캡슐화는 어떻게 높은 응집도와 낮은 결합도를 갖게 할까요? 정보 은닉을 활용합니다. 외부에서 접근할 필요가 없는 것들은 private으로 접근하지 못하도록 제한을 두는 것입니다. (객체안의 필드를 선언할 때 private으로 선언하라는 말이 바로 이 때문!!)	상	
DNS란 무엇일까요	DNS (Domain Name Server) 모든 통신은 IP를 기반으로 연결되지만 사용자에게 일일히 IP 주소를 입력하기란 UX적으로 좋지 않습니다. 때문에 DNS 가 등장 했으며 DNS 는 IP 주소와 도메인 주소를 매핑하는 역할을 수행합니다.	상	
도메인 주소가 IP로 변환되는 과정은?	**도메인 주소가 IP로 변환되는 과정** 1. 디바이스는 hosts 파일을 열어 봅니다 - hosts 파일에는 로컬에서 직접 설정한 호스트 이름과 IP 주소를 매핑 하고 있습니다 2. DNS는 캐시를 확인 합니다 - 기존에 접속했던 사이트의 경우 캐시에 남아 있을 수 있습니다 - DNS는 브라우저 캐시, 로컬 캐시(OS 캐시), 라우터 캐시, ISP(Internet Service Provider)캐시 순으로 확인 합니다 3. DNS는 Root DNS에 요청을 보냅니다 - 모든 DNS에는 Root DNS의 주소가 포함 되어 있습니다 - 이를 통해 Root DNS에게 질의를 보내게 됩니다 - Root DNS는 도메인 주소의 최상위 계층을 확인하여 TLD(Top Level DNS)의 주소를 반환 합니다 4. DNS는 TLD에 요청을 보냅니다 - Root DNS로 부터 반환받은 주소를 통해 요청을 보냅니다 - TLD는 도메인에 권한이 있는 Authoritative DNS의 주소를 반환 합니다 5. DNS는 Authoritative DNS에 요청을 보냅니다 - 도메인 이름에 대한 IP 주소를 반환 합니다 - 이때 요청을 보내는 DNS의 경우 재귀적으로 요청을 보내기 때문에 `DNS 리쿼서`라 지칭 하고 요청을 받는 DNS를 `네임서버`라 지칭 합니다	중	