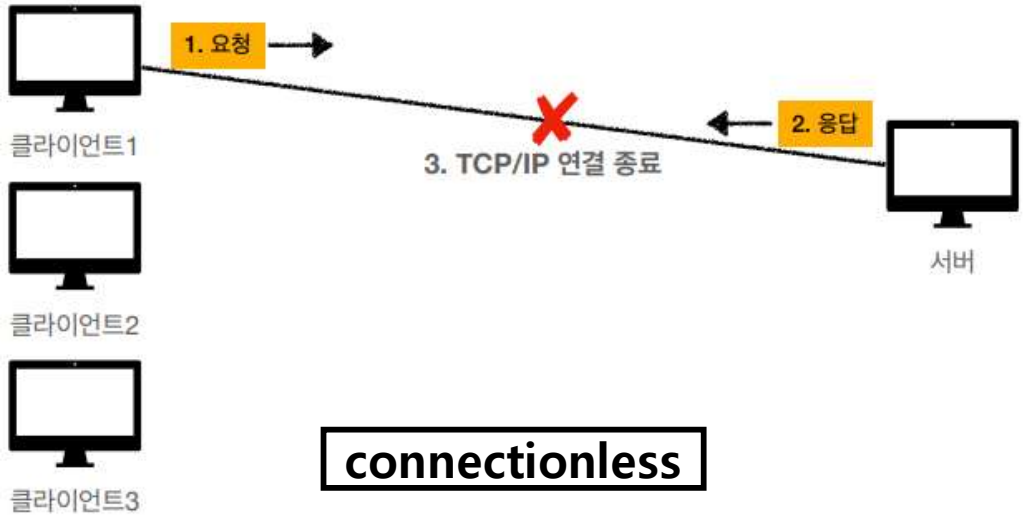
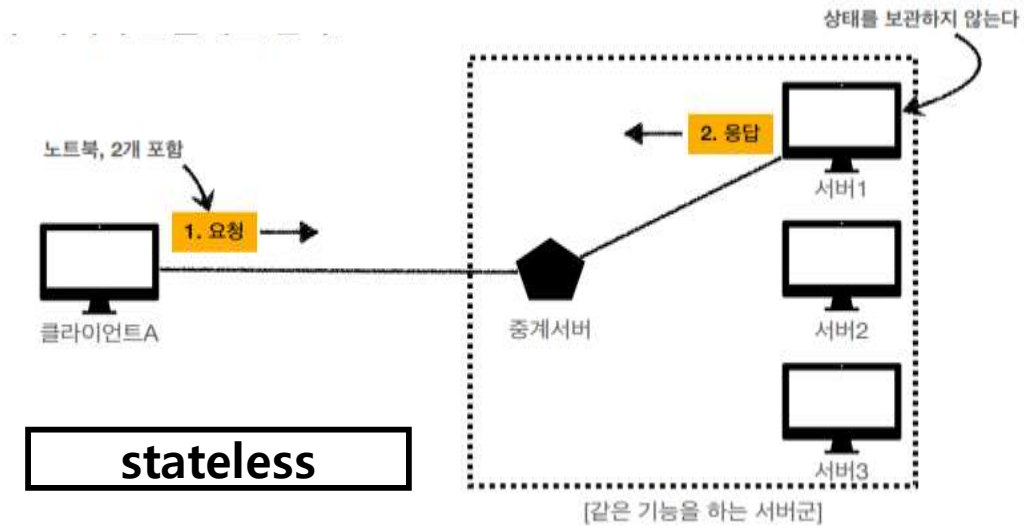
쿠키와 세션

발표 자료

쿠키의 등장

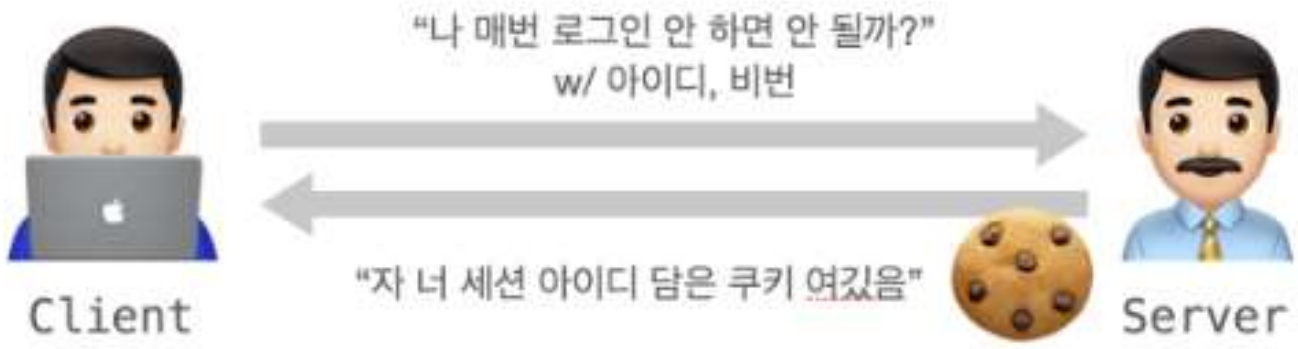
- HTTP 특징
  - Connectionless
    - 클라이언트가 요청 보내고, 서버가 응답 보내면 바로 연결 끊음.
  - Stateless
    - 연결 끊는 순간 클라이언트와 서버의 통신 끝나고 상태 정보 유지 X.





쿠키와 세션 필요성

- HTTP 특징 때문에 현재 접속 사용자가 이전 접속 사용자인지 알 수 없음.
- 통신할 때마다 새로 연결해 클라이언트는 매 요청마다 인증 해야 함.
- HTTP에서 상태 유지하기 위한 기술로 쿠키, 세션 사용

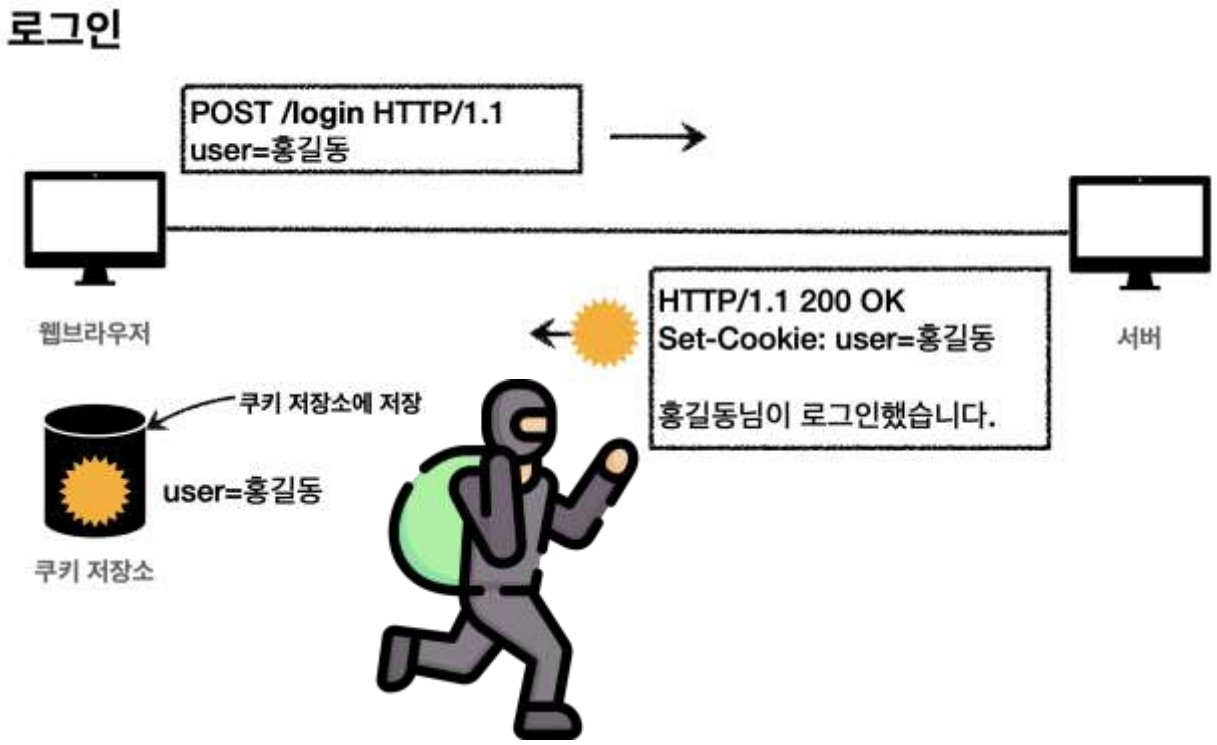


쿠키 개념

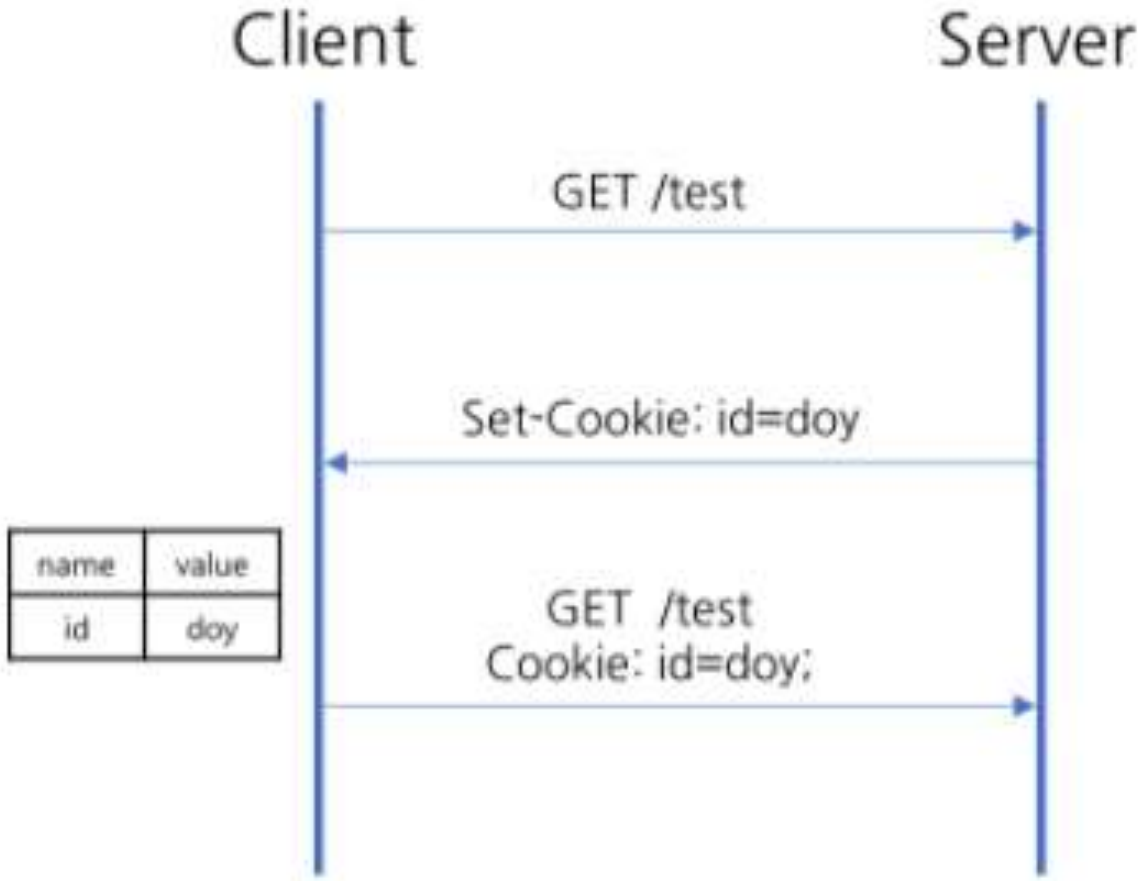
- 클라이언트 로컬에 저장되는 파일로 클라이언트 측에서 관리.
- 쿠키 저장했다가 동일 서버에 재 요청 시 저장된 데이터와 함께 전송.
- 사용자 인증이 유효한 시간을 명시할 수 있음.
- 유효 시간이 정해지면 브라우저가 종료되어도 인증이 유지됨.
- 사용자 정보 컴퓨터에 남아 보안에 취약.

쿠키 단점

- 보안에 취약
- 용량이 작음
- 브라우저마다 지원 형태 다름
- 한번에 하나의 정보만 저장 가능
- 문자열 그대로 통신해 위변조 가능
- 네트워크 부하 심해짐



쿠키 동작 방식



1. 클라이언트가 서버에 요청
2. 서버에서 쿠키 생성
3. 응답 헤더에 쿠키 넣어 전송
4. 받은 쿠키 클라이언트는 저장
5. 다시 서버에 요청 시 헤더에 쿠키 넣어 전송
6. 서버는 쿠키 보고 클라이언트 식별 및 이전 상태 확인

세션 개념

- 쿠키의 단점인 보안적 이슈 해결 위해 세션 등장
- 방문자가 웹 서버에 접속해 있는 상태가 세션
- 쿠키와 달리 웹 서버에 상태 정보 저장해 보안 좋음
- 각 클라이언트에 고유 세션 ID 부여해 상태 유지
- 클라이언트는 세션 ID를 쿠키 형태로 가짐
- 클라이언트는 세션 ID 이용해 서버의 데이터 사용

세션 동작 방식

1. 클라이언트가 요청➡ **2.** 서버는 클라이언트의 헤더에서 쿠키 확인해 세션 ID 확인 ➡

3. 세션 ID 없다면 서버는 세션 ID 부여➡ **4.** 서버 응답 시 헤더에 세션 ID 포함한 쿠키 넣어 전송

➡ **5.** 클라이언트는 세션 ID 있는 쿠키 저장➡ **6.** 브라우저 닫기 전 재 요청 시 헤더에 세션 ID

포함된 쿠키 넣어 전송➡ **7.** 서버는 세션 ID 확인 후 응답➡ **8.** 브라우저 종료 시 세션 정보 삭제

세션 장단점

장점

- 쿠키 보단 보안이 좋음
- 세션 ID 자체는 유의미한 개인 정보 X
- 서버 용량 허용 저장 데이터 제한 X
- 서버에 저장해 관리 편함
- 고유 세션 ID 발급되어 요청 시 회원 정보 확인 필요 X

단점

- 세션 ID로 서버 접근 시 사용자 정보 접근 가능
- 서버에서 클라이언트 상태 유지해 서버 메모리 차지 -> 서버 과부화, 성능 저하
- 서버 확장 시 분산 처리해야 해서 확장 어려움

쿠키 vs 세션

	쿠키	세션
저장 위치	클라이언트 PC (브라우저)	웹 서버
저장 형식	Text	Object
만료 시점	쿠키 저장 시 설정 브라우저 종료되어도 만료 시점 지나지 않으면 자동 삭제 X	브라우저 종료 시 삭제 기간 지정 가능 but 브라우저 종료 시 무조건 삭제
용량 제한	총 300개 하나의 도메인당 20개 하나의 쿠키 당 4KB	서버가 허용하는 한 용량 제한 X
속도	클라이언트에 저장되어 요청 시 세션보다 빠름	정보가 서버에 있어 서버 처리 필요해 쿠키보다 느림
보안	클라이언트에 저장되어 변질, 위조 될 수 있어 세션보다 취약	세션 ID만 저장하고 서버에서 처리해 쿠키보다 좋음