빠른 웹사이트들의 필수 스택 10개

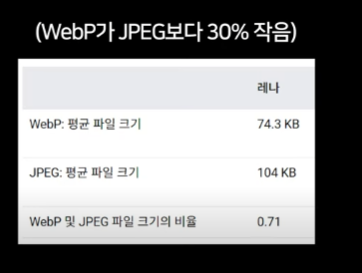
1. 압축

-> react 를 사용하게 되면 용량이 커짐. 이를 그대로 보내지 않고 유저에게 압축해서 보냄.





2. 이미지 포맷



Gif -> webP 나 mp4 로 변환.



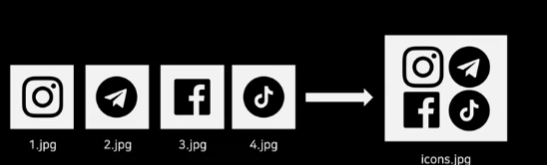
영상 압축 알고리즘도 많이 생김.

3. HTTP2 & HTTP3

유저가 사이트 접속 시 HTML 을 먼저 받고, 이후 HTML에 있는 js나 css와 같은 파일들을 요청하게 됨. 이때 HTTP 방식은 한 번에 최대 6개까지 전송 가능하다.

HTTP2 방식은 한번의 tcp연결로 여러 개의 파일을 병렬적으로 가져올 수 있음.

예전엔 이미지를 하나로 합쳐서 스프라이트 이미지를 만들어서 보냄



또한 css를 여러 개를 합쳐서 하나의 파일로 보냄.

HTTP3 방식부터는 tcp 말고 udp방식을 사용하기 때문에 tcp가 가지는 전송 실패 시 전체 중단이 없음. 모바일 환경에서도 유저가 ip가 변경되어도 계속 연결 유지됨.

4. 핑줄이기

해외에 있는 곳에서 접속하면 지연시간 핑이 생김. 이를 해결하기 위해 해외 곳곳에 서버를 퍼뜨려놓고 유저들이 가장 가까운 곳에서 가져올 수 있도록 하면 됨. -> CDN 쓰면 됨.



5. DB안쓰기

유저가 많아져서 입출력 요청이 많거나 락이 걸리면 느려질 수 있음.

컴퓨터의 사양을 높이거나, db를 여러대 두는 방식으로 해결할 수 있지만, 가장 좋은 방법은 db를 최대한 안쓰는 것이다. 이를 위해 db 출력 결과를 메모리에 넣고, 요청이 올 때마다 이를 전달하는 방식을 사용할 수 있음. (캐싱 - 메모리 데이터 베이스를 쓰기도 함. Redis)

정확도가 그렇게 중요하지 않는 데이터를 입력하는 경우엔 여러 요청을 모아서 한 번에 전달하는 것도 방법이다.

6. 번들링

프론트 엔드에서는 여러 파일을 하나로 합치는 번들링을 사용하기도 함. (Vite)

- > 코드 다이어트 가능. 안쓰는 것들을 싹 날릴 수 있음.

7. 캐시컨트롤

웹페이지에 방문하면 자동으로 캐싱을 해둠. Css js img와 같은 파일을 첫 접속 시에 하드에 담아두고, 재요청 시에 이를 전달함. 대부분 자동인데 캐시 컨트롤 옵션을 만져서 캐시 기간을 정할 수 있음.

8. 마지막 수정일

서버에서 파일을 유저에게 보내줄 때 last-modified라는 이름으로 언제 마지막으로 수정됐는지를 알려줄 수 있음. 다시 방문할 때 캐싱 기간이 지났을 대 이를 재요청하게 됨. 서버에서 마지막 수정날짜 이후로 변경된 것이 있으면 새로운 것을 주고 아니라면 304를 보냄.

9. 14KB

웹페이지를 14kb이하로 만들어두는 것도 좋은 방법임. tcp방식을 할 때 14kb부터 전송한 뒤 서서히 속도가 올라가기 때문.

10. Prefetch

href를 통해서 페이지 미리 로드해둘 수 있게 할 수도 있음.