원티드 프론트엔드 챌린지
8월 3회차

Recap

1. 테스트는 비지니스 로직을 테스트한다
    a. UI는 비지니스 로직과 관련된 것만 테스트
    b. style은 테스트 하지 않는다
2. 테스트케이스 1개당 하나의 로직만 테스트한다
    a. describe안에 it/test를 사용하여 하나의 로직을 테스트
    b. 각 테스트 케이스는 `expect`를 사용해서 검증한다
        i. 실제 컴포넌트에서 성공한 경우 어떤 일이 일어나는지
        ii. 실제 컴포넌트에서 실패한 경우 어떤 일이 일어나는지
        iii. 활용하여 검증
3. 테스트코드 리뷰

Agenda

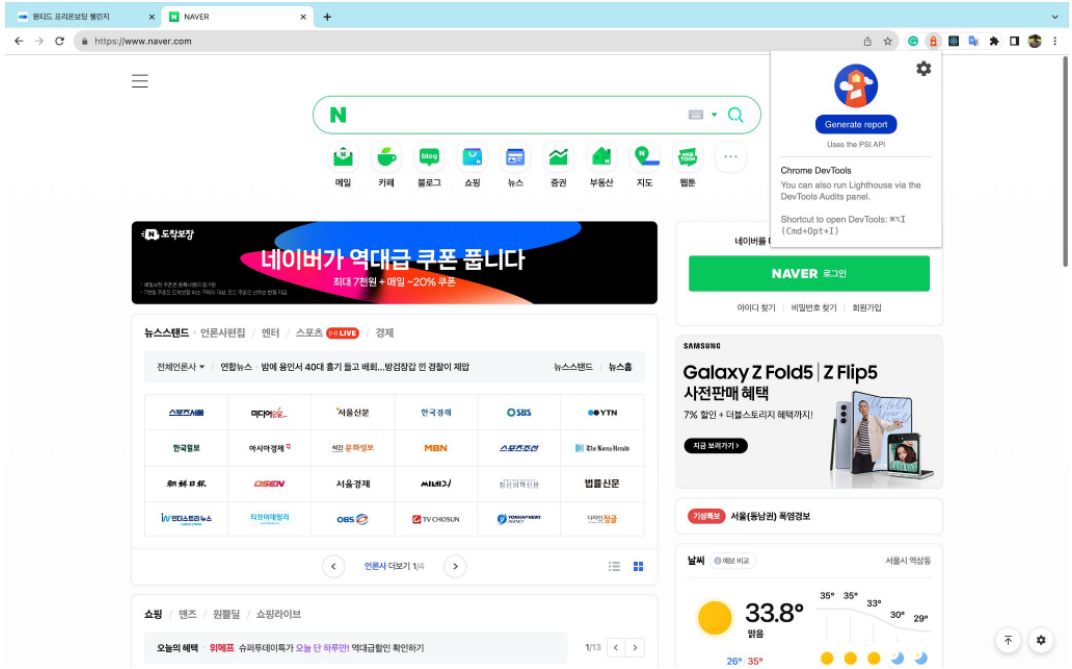
1. 성능 측정 툴 소개 및 사용법
    a. Lighthouse
    b. Performance
    c. Profiler
2. 최적화
    a. Code Splitting
    b. Lazy Loading

Lighthouse

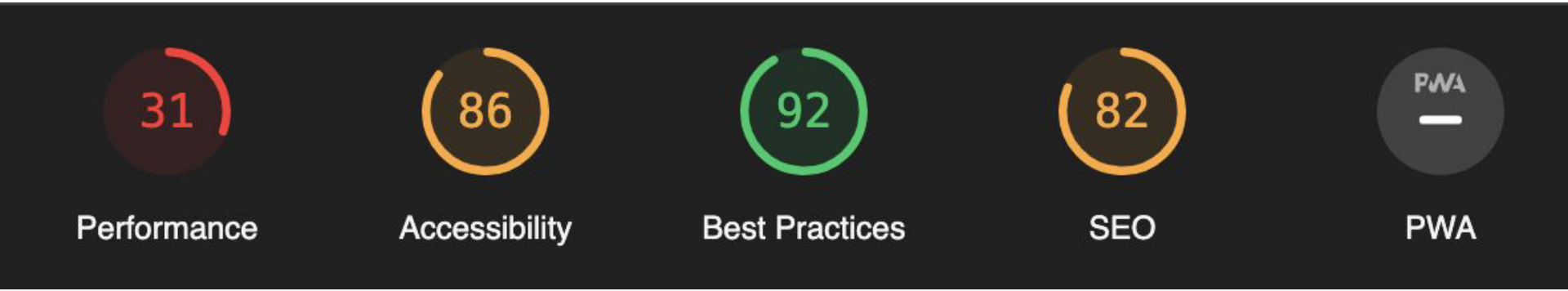
1. 웹 성능을 분석하는 오픈소스
2. 주요기능
    a. 성능분석
    b. 접근성검사
    c. SEO평가
    d. PWA기준 평가

Lighthouse

1. Chrome extension 활용 가능



Lighthouse



1. Performance - 성능
2. Accessibility - 접근성
3. Best Practices - 보안관련
4. SEO - 검색 최적화
5. PWA - PWA 기준 확인

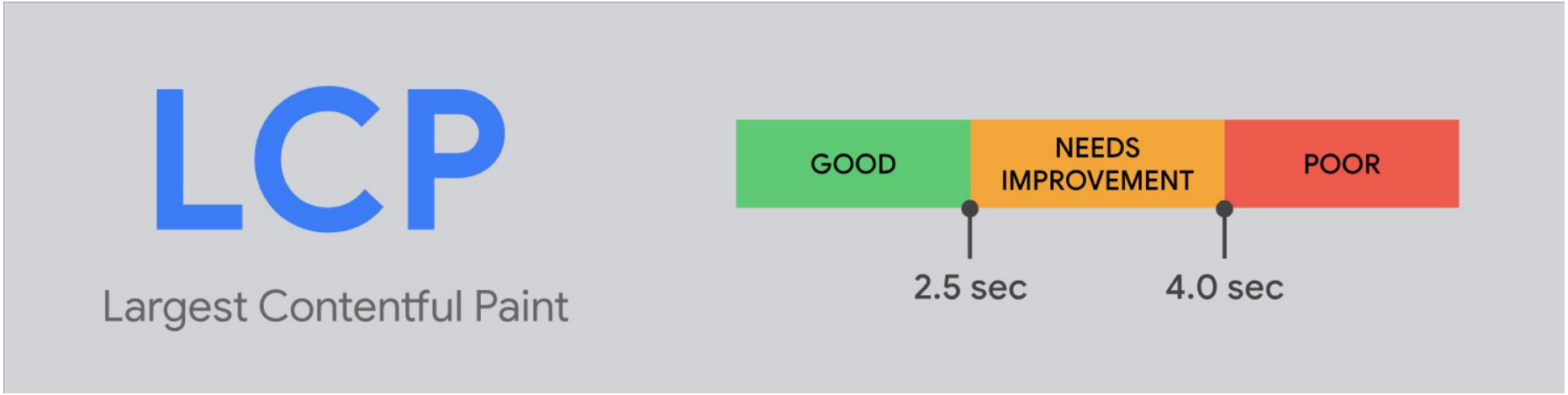
Web Vitals

Overview

Web Vitals is an initiative by Google to provide unified guidance for quality signals that are essential to delivering a great user experience on the web.

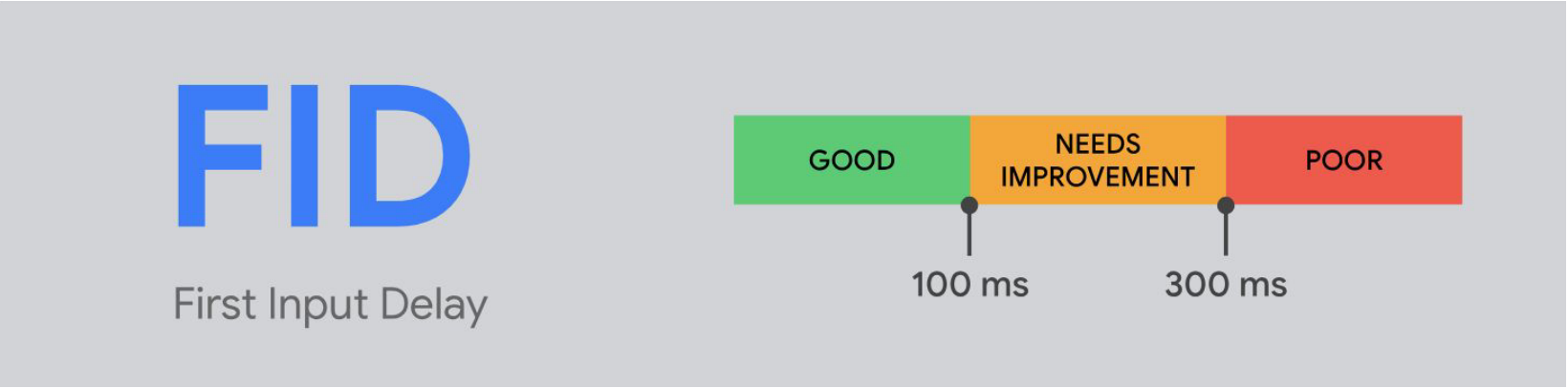
1. 사용자 경험을 측정하는 수치

Web Core Vitals - LCP



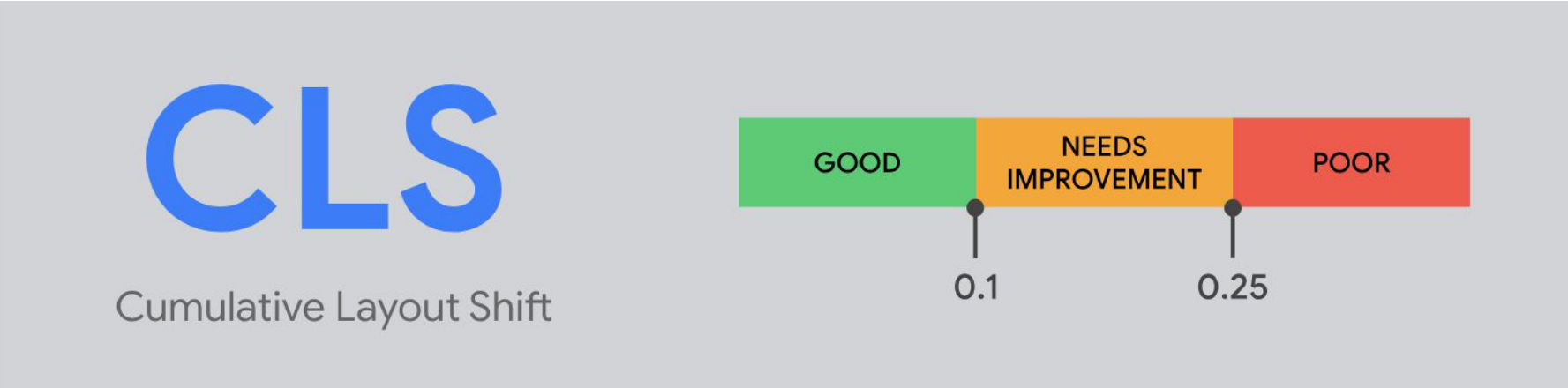
1. Largest Contentful Paint
2. 페이지의 `주요 콘텐츠`가 얼마나 빨리 로드되는지 측정
    a. `주요 콘텐츠`라는 기준이 애매함
    b. 실제로는 가장 큰 텍스트나 이미지를 나타냄
    c. `빈칸이 언제 없어지는가` 정도로 표현 가능

Web Core Vitals - FID



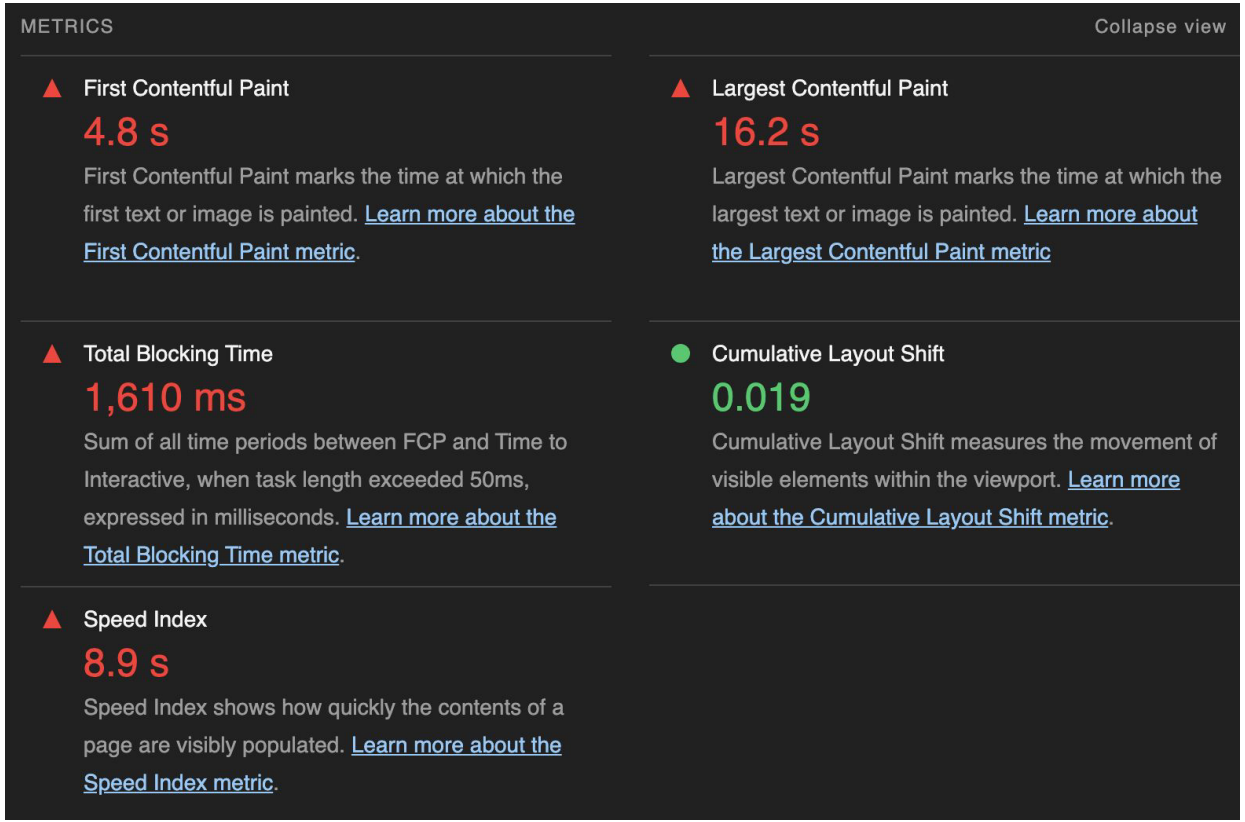
1. First Input Delay
2. 첫 이벤트 핸들링에 소요되는 시간
   a. 사용자가 버튼을 클릭하면
   b. 해당 버튼의 이벤트가 언제 발생하는지?

Web Core Vitals - CLS



1. Cumulative Layout Shift
2. 설계 이슈로 UI가 변경되는 것
3. 영상참고
    a. https://web.dev/cls/

기타 Web Vitals



Performance Tab

1. Runtime 성능 측정
    a. 렌더링 성능
    b. 자바스크립트 성능
    c. 메모리 관리
    d. 반응성(First Input Delay)
    e. 네트워크 성능
2. 주요 정보
    a. Loading
    b. Scripting
    c. Rendering
    d. Painting

Performance Tab

1. Memory
    a. JS Heap
    b. Documents
    c. Nodes
    d. Listeners
    e. GPU Memory

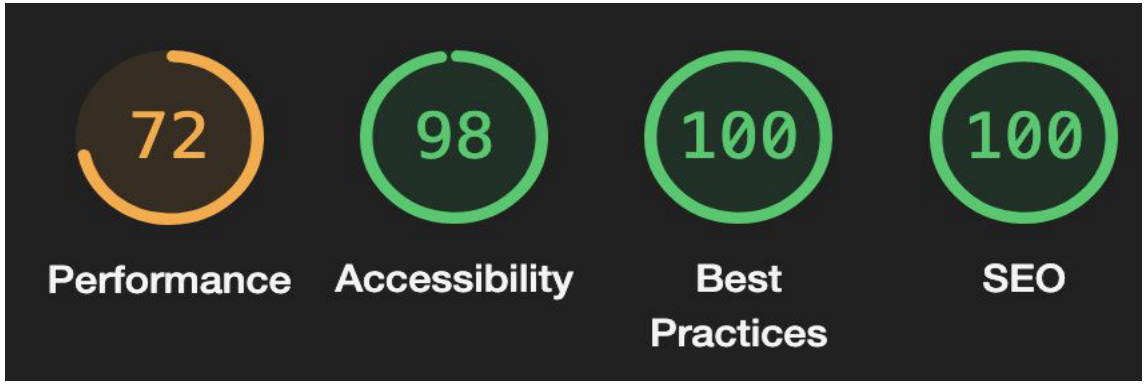
Profiler

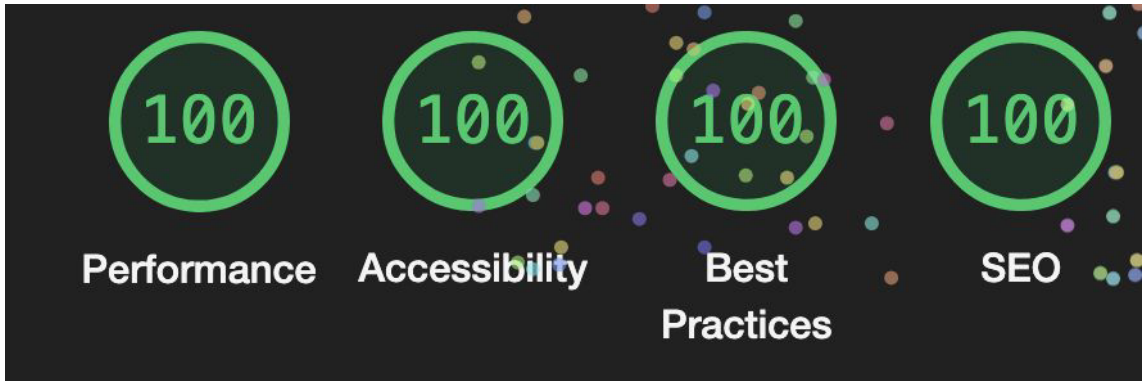
1. React 내장기능
   a. https://react.dev/reference/react/Profiler
2. 컴포넌트 별 렌더링 시간 측정 가능
3. 특정 컴포넌트에서 병목이 발생한다면 해결

Lighthouse 사용법

1. Lighthouse는 production build에서 확인
    a. Production build 시 나름의 최적화가 진행되어 web vital이 상당부분 개선됨
        i. Minification
        ii. Tree shaking

Lighthouse 사용법





Minification

1. 빈칸, 줄바꿈 등을 제거함
2. Vite 기준 dist/asset/.js

```
function MS(e,t){for(var n=0;n<t.length;n++){const r=t[n];if(typeof r!="string"&&
!Array.isArray(r)){for(const o in r)if(o!=="default"&&!(o in e)){const i=Object.
getOwnPropertyDescriptor(r,o);i&&Object.defineProperty(e,o,i.get?i:{enumerable:!0
get:()=>r[o]})}}}return Object.freeze(Object.defineProperty(e,Symbol.toStringTag,
{value:"Module"}))}(function(){const t=document.createElement("link").relList;if(
```

Tree Shaking

1. Lighthouse는 production build에서 확인
   a. Production build 시 나름의 최적화가 진행되어 web vital이 상당부분 개선됨
      i. Minification
      ii. Tree shaking
2. dev에서는 불필요한 내용들이 많이 보임
   a. 설치한 라이브러리에서 뭘 수정해라…
   b. 이걸 반영하려면 모든 라이브러리를 fork해서 수정해야함

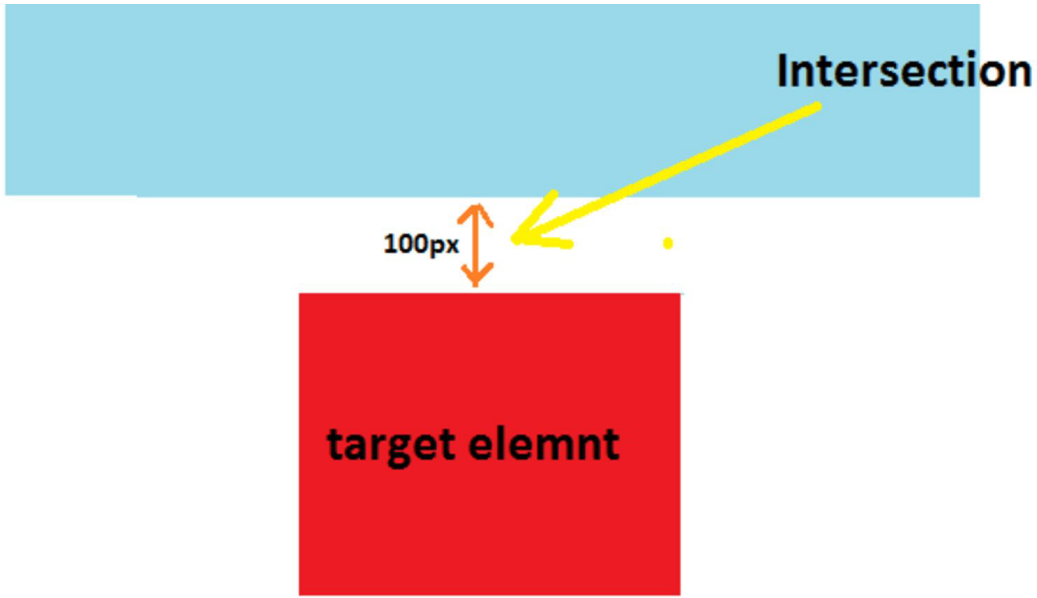
…deps/@tanstack_react-query-devtools.js?v=fb6e4e18 (localhost)	98.0 KiB	51.5 KiB
…@tanstack/react-query-devtools/src/devtools.tsx	17.6 KiB	8.6 KiB
…superjson/src/transformer.ts	6.5 KiB	5.1 KiB
…@tanstack/match-sorter-utils/src/index.ts	5.0 KiB	4.8 KiB
…@tanstack/react-query-devtools/src/Explorer.tsx	5.3 KiB	4.7 KiB
…superjson/src/plainer.ts	4.7 KiB	4.7 KiB

최적화 - Code Splitting

1. Bundle size를 줄이는 방법
    a. 전체 용량이 줄어드는 것은 아니고 하나의 번들을 여러개로 잘게 쪼개는 것
    b. React lazy + Suspense의 조합을 사용함
        i. https://web.dev/code-splitting-suspense/
        ii. https://react.dev/reference/react/lazy
        iii. https://react.dev/reference/react/Suspense
2. 최초 로딩시간이 단축되는 이점이 있음
    a. FCP, LCP, FID 개선 가능
    b. https://jasonkang14.github.io/react/optimzation-with-chat-gpt
3. 브라우저 cache

Lazy Loading

1. Lighthouse 만점이라고 최적화가 끝난것이 아님
2. IntersectionObserver



[아하!모먼트] 개발을 공부하는 나만의 방법이 있나요?

1. 블로그
2. 사이드 프로젝트
3. 스터디
4. 컨퍼런스
5. 커뮤니티