

포트폴리오 작성을 위한 리액트
프로젝트 고도화
3회차

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

배포, production 환경에서만 적용해야함
시크릿모드에서는 더 빠르다고 함

1. Chrome extension 활용 가능

The screenshot shows a Chrome browser window with the Naver homepage loaded. The Lighthouse Chrome extension sidebar is open on the right side of the browser. The sidebar contains the following elements:

- A "Generate report" button with the text "Uses the PSI API".
- A section for "Chrome DevTools" with the text: "You can also run Lighthouse via the DevTools Audit panel. Shortcut to open DevTools: *⌘I (Cmd+Opt+I)".
- A green "NAVER 로그인" button.
- Links for "아이디 찾기", "비밀번호 찾기", and "회원가입".
- A section for "SAMSUNG" with the text "Galaxy Z Fold5 | Z Flip5 사전판매 혜택" and "7% 할인 + 더블스토리지 혜택까지". A button "지금 보러가기" is present.
- A section for "기초특보" with the text "서울(동남권) 폭염경보".
- A weather section for "날씨" showing "33.8° 맑음" and a forecast for the next few days.

The Naver homepage in the background includes a search bar, navigation links (메일, 카페, 블로그, 소인, 뉴스, 증권, 부동산, 지도, 웹툰), a banner for "네이버가 역대급 쿠폰 풀니다" (Naver is releasing record-breaking coupons), and a news section titled "뉴스스탠드" with various news categories and a table of links.

Lighthouse



Performance



Accessibility



Best Practices



SEO



PWA

1. Performance - 성능
2. Accessibility - 접근성 story book 에서 보는게 더 정확함
3. Best Practices - 보안관련
4. SEO - 검색 최적화
5. PWA - PWA 기준 확인

Web Vitals

Overview

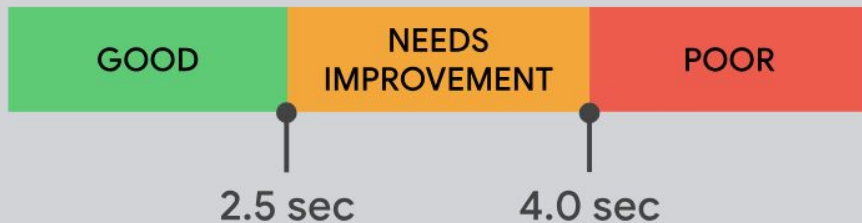
Web Vitals is an initiative by Google to provide unified guidance for quality signals that are essential to delivering a great user experience on the web.

1. 사용자 경험을 측정하는 수치

Web Core Vitals - LCP

LCP

Largest Contentful Paint



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

FID를 2024년 3월에 INP(Interaction to Next Paint)로 교체할 예정이라고 합니다

FID

First Input Delay



1. First Input Delay total blocking time 이랑 비슷한 맥락
2. 첫 이벤트 핸들링에 소요되는 시간
 - a. 사용자가 버튼을 클릭하면
 - b. 해당 버튼의 이벤트가 언제 발생하는지?

Web Core Vitals - CLS

CLS

Cumulative Layout Shift



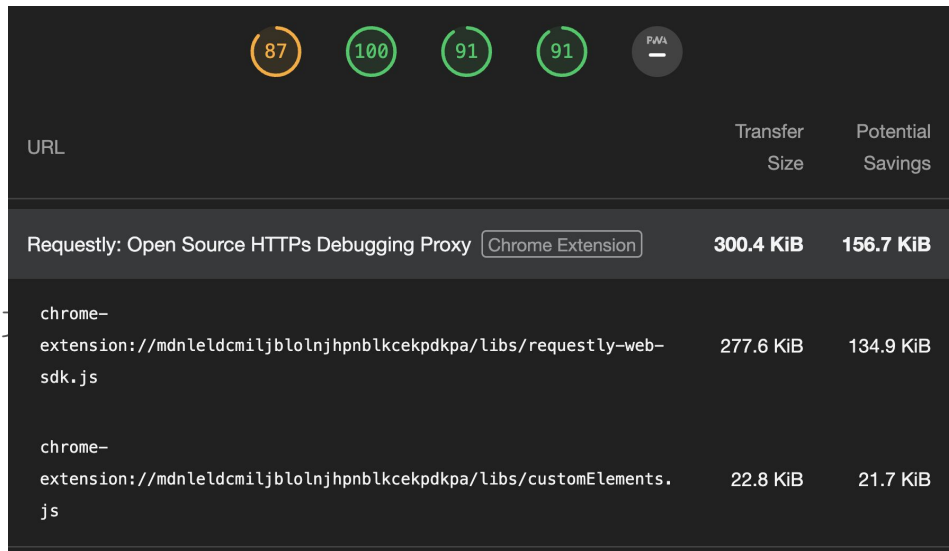
1. Cumulative Layout Shift
2. 설계 이슈로 UI가 변경되는 것
3. 영상참고
 - a. <https://web.dev/cls/>

기타 Web Vitals

METRICS		Collapse view
▲ First Contentful Paint	▲ Largest Contentful Paint	
4.8 s	16.2 s	
First Contentful Paint marks the time at which the first text or image is painted. Learn more about the First Contentful Paint metric.	Largest Contentful Paint marks the time at which the largest text or image is painted. Learn more about the Largest Contentful Paint metric	
▲ Total Blocking Time	● Cumulative Layout Shift	
1,610 ms	0.019	
Sum of all time periods between FCP and Time to Interactive, when task length exceeded 50ms, expressed in milliseconds. Learn more about the Total Blocking Time metric.	Cumulative Layout Shift measures the movement of visible elements within the viewport. Learn more about the Cumulative Layout Shift metric.	
▲ Speed Index		
8.9 s		
Speed Index shows how quickly the contents of a page are visibly populated. Learn more about the Speed Index metric.		

Lighthouse 주의사항

1. Production build에서 테스트
 - a. 가급적 Incognito에서 테스트
 2. `node_modules`내에서 발생하는 에러들은 무시
 - a. 정말 해결하고 싶다면 해당 패키지 제
- cache 종류=> disk(참고), memory(가방)



The screenshot shows the Lighthouse interface with four performance scores: 87 (orange), 100 (green), 91 (green), and 91 (green). A PWA icon is also visible. Below the scores is a table of audits. The first audit is for 'Requestly: Open Source HTTPs Debugging Proxy' (Chrome Extension) with a transfer size of 300.4 KiB and a potential saving of 156.7 KiB. Below it are two smaller audits for 'chrome-extension://mdnlldcmiljblolnjhpnblkcekpdka/libs/requestly-web-sdk.js' (277.6 KiB, 134.9 KiB saving) and 'chrome-extension://mdnlldcmiljblolnjhpnblkcekpdka/libs/customElements.js' (22.8 KiB, 21.7 KiB saving).

URL	Transfer Size	Potential Savings
Requestly: Open Source HTTPs Debugging Proxy Chrome Extension	300.4 KiB	156.7 KiB
chrome-extension://mdnlldcmiljblolnjhpnblkcekpdka/libs/requestly-web-sdk.js	277.6 KiB	134.9 KiB
chrome-extension://mdnlldcmiljblolnjhpnblkcekpdka/libs/customElements.js	22.8 KiB	21.7 KiB

프론트 만의 문제가 아닌것이 => 네트워크가 빠르면 lighthouse 가 높게 나오는 경우도 있음

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

Profiler

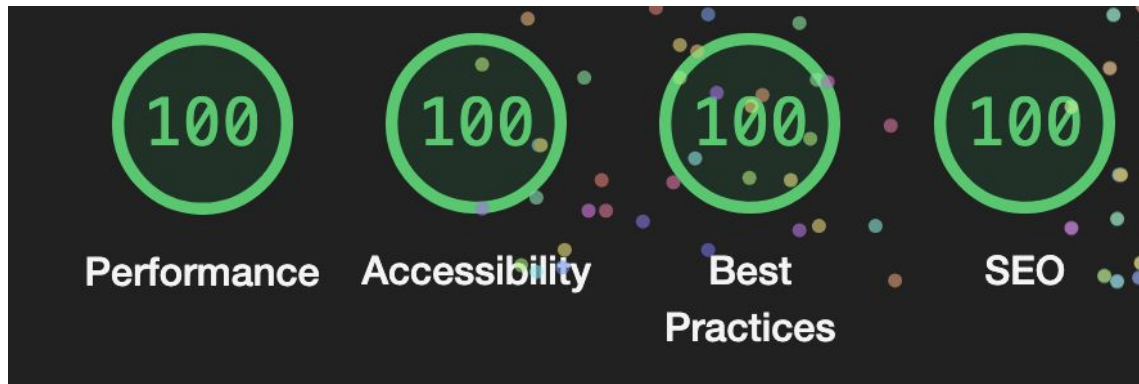
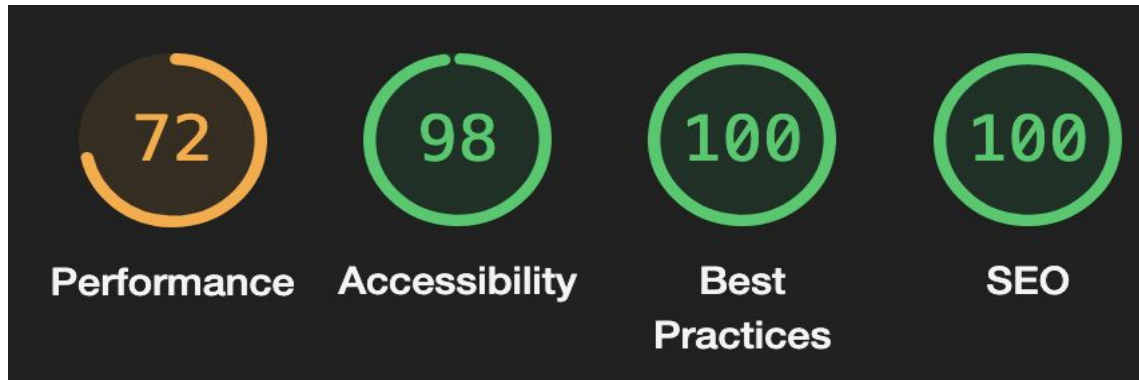
1. React 내장기능
 - a. <https://react.dev/reference/react/Profiler>
2. 컴포넌트 별 렌더링 시간 측정 가능
3. 특정 컴포넌트에서 병목이 발생한다면 해결

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

프로젝트가 클때 적용해야 좋음

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 hash 를 기준으로 browser 에서 캐싱을 하기때문에 더 이득이됨
- b. <https://jasonkang14.github.io/react/optimzation-with-chat-gpt>

3. 브라우저 cache

Lazy Loading

1. Lighthouse 만점이라고 최적화가 끝난것이 아님
2. IntersectionObserver



무한 스크롤 같은 경우 스크롤이 바닥에 닿을때 데이터가 로드될수 있게 하자

<https://ko.legacy.reactjs.org/docs/profiler.html#gatsby-focus-wrapper>

최적화에서 중요한 것

1. 어떤 최적화를 했느냐는 중요하지 않음
 - a. Code-splitting 적용
 - b. Lazy loading 적용
 - c. 위 항목들 자체로는 무의미하고, "왜" 해당 방식으로 최적화 했는지 설명할 수 있어야 함

병목을 찾고 에러를 해결하는것

최적화에서 중요한 것

1. 어떤 최적화를 했느냐는 중요하지 않음
 - a. Code-splitting 적용
 - b. Lazy loading 적용
 - c. 위 항목들 자체로는 무의미하고, "왜" 해당 방식으로 최적화 했는지 설명할 수 있어야 함
2. 최적화의 의미는 코드의 문제점을 파악하고 해결하는 것

최적화에서 중요한 것

1. 어떤 최적화를 했느냐는 중요하지 않음
 - a. Code-splitting 적용
 - b. Lazy loading 적용
 - c. 위 항목들 자체로는 무의미하고, "왜" 해당 방식으로 최적화 했는지 설명할 수 있어야 함
2. 최적화의 의미는 코드의 문제점을 파악하고 해결하는 것
 - a. 문제해결 능력을 어필하는 식으로 이력서를 작성해야함

최적화에서 중요한 것

1. 어떤 최적화를 했느냐는 중요하지 않음
 - a. Code-splitting 적용
 - b. Lazy loading 적용
 - c. 위 항목들 자체로는 무의미하고, "왜" 해당 방식으로 최적화 했는지 설명할 수 있어야 함
2. 최적화의 의미는 코드의 문제점을 파악하고 해결하는 것
 - a. 문제해결 능력을 어필하는 식으로 이력서를 작성해야함
 - b. `react-router-dom routerConfig`에 lazy loading 적용

최적화에서 중요한 것

1. 어떤 최적화를 했느냐는 중요하지 않음
 - a. Code-splitting 적용
 - b. Lazy loading 적용
 - c. 위 항목들 자체로는 무의미하고, "왜" 해당 방식으로 최적화 했는지 설명할 수 있어야 함
2. 최적화의 의미는 코드의 문제점을 파악하고 해결하는 것
 - a. 문제해결 능력을 어필하는 식으로 이력서를 작성해야함
 - b. react-router-dom routerConfig에 lazy loading 적용
 - c. Lazy loading 적용으로 index.js bundle size 감소로 FCP 60% 개선