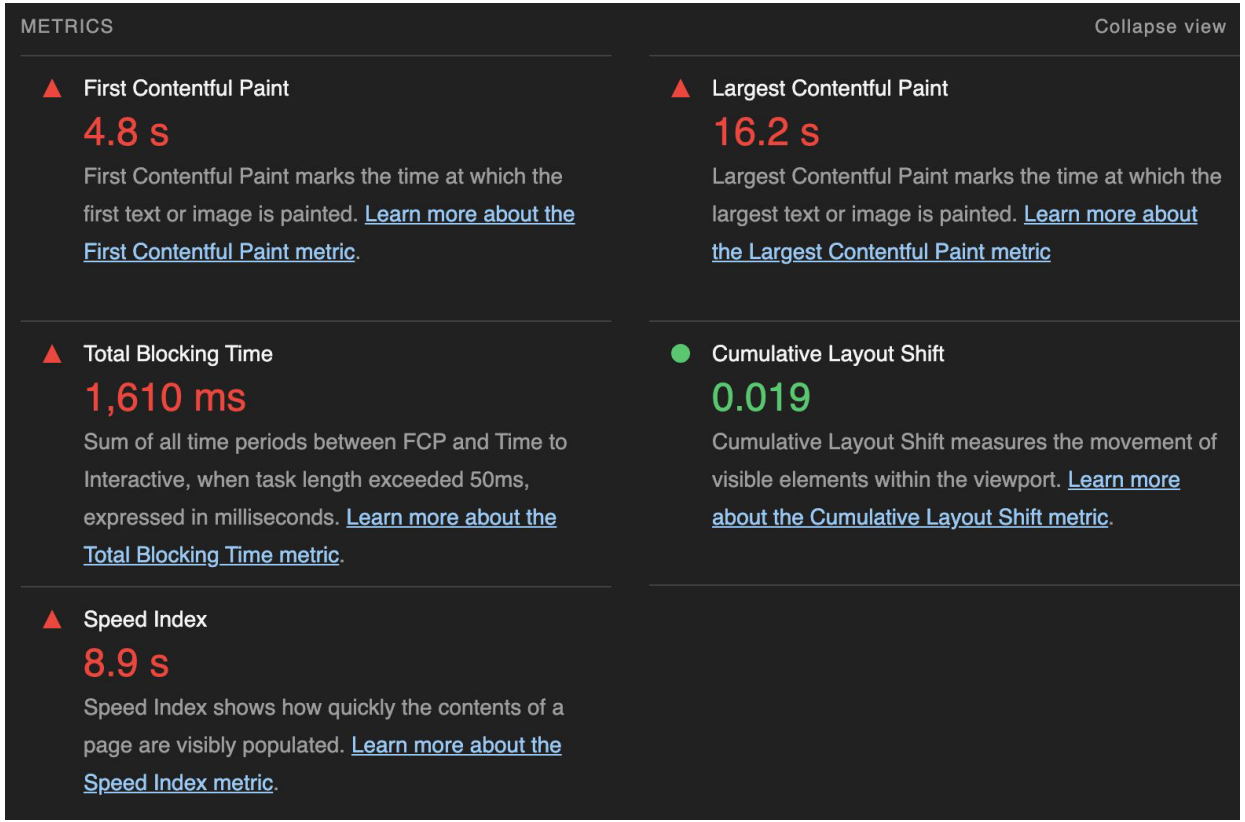
원티드 프론트엔드 챌린지
8월 4회차

Recap

1. 최적화 관련 툴
    a. Lighthouse
    b. Performance
    c. Profiler
2. 최적화 방안
    a. Code Splitting
    b. npm dedupe
    c. Lazy Loading

Lighthouse


METRICS
Collapse view
First Contentful Paint
4.8 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
16.2 s
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
1,610 ms
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
0.019
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.

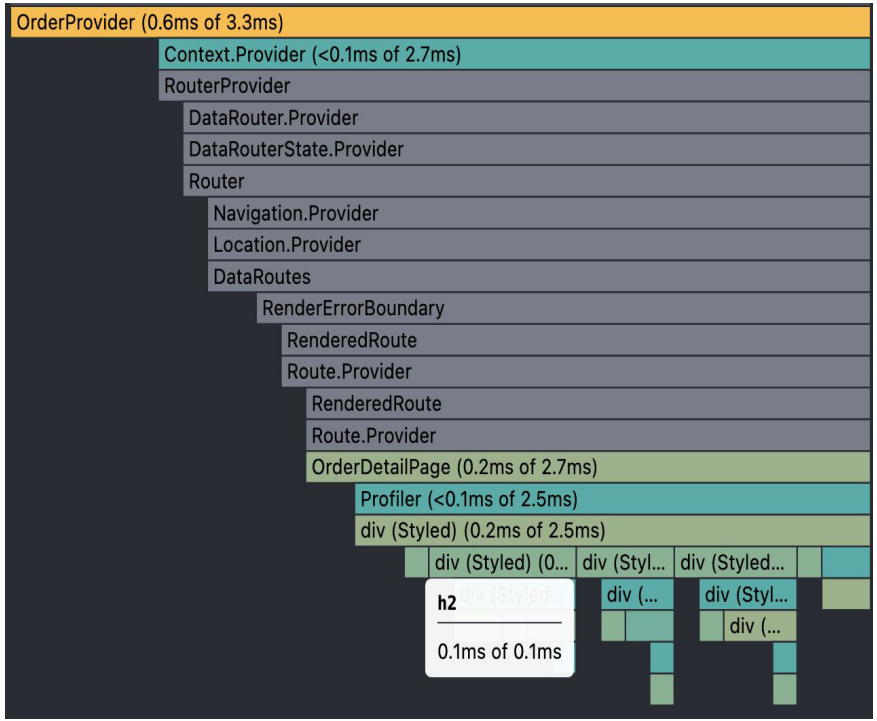
Lighthouse

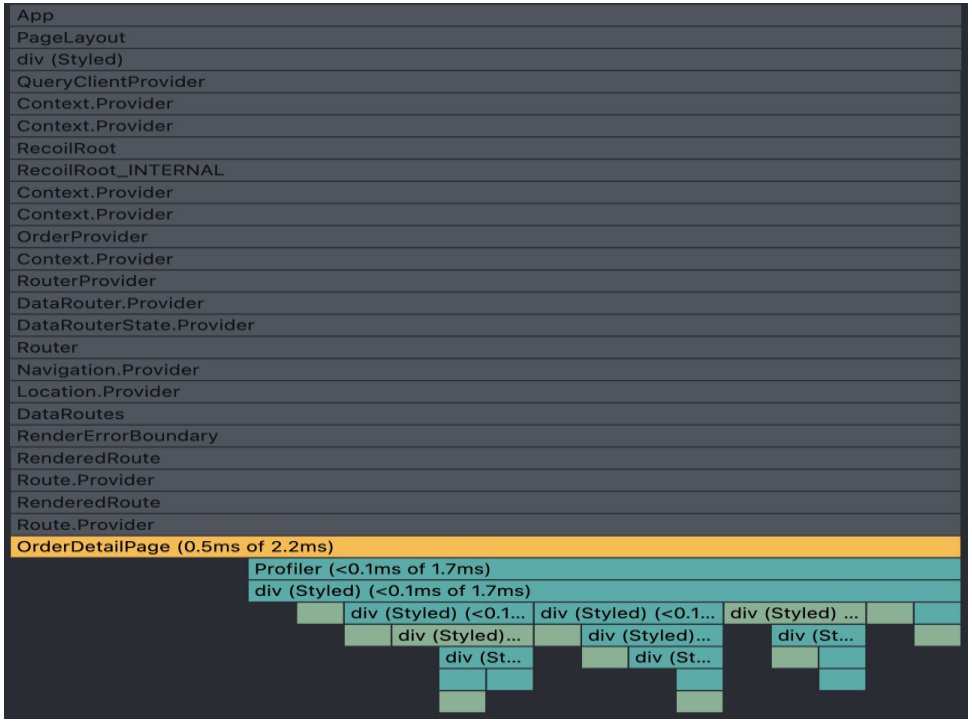
1. Lighthouse는 production build에서 확인
    a. Production build 시 나름의 최적화가 진행되어 web vital이 상당부분 개선됨
        i. Minification
        ii. Tree shaking

Performance

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


OrderProvider (0.6ms of 3.3ms)
Context.Provider (<0.1ms of 2.7ms)
RouterProvider
DataRouter.Provider
DataRouterState.Provider
Router
Navigation.Provider
Location.Provider
DataRoutes
RenderErrorBoundary
RenderedRoute
Route.Provider
RenderedRoute
Route.Provider
OrderDetailPage (0.2ms of 2.7ms)
Profiler (<0.1ms of 2.5ms)
div (Styled) (0.2ms of 2.5ms)
div (Styled) (0...
div (Styl...
div (Styled...
div (...
div (Styl...
div (...
h2
0.1ms of 0.1ms

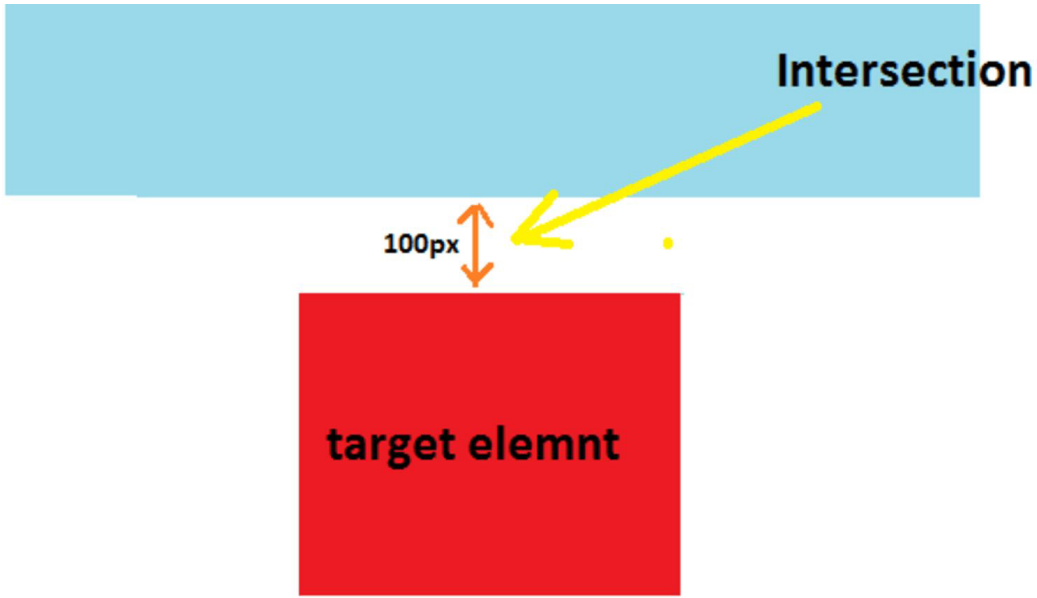

App
PageLayout
div (Styled)
QueryClientProvider
Context.Provider
Context.Provider
RecoilRoot
RecoilRoot_INTERNAL
Context.Provider
Context.Provider
OrderProvider
Context.Provider
RouterProvider
DataRouter.Provider
DataRouterState.Provider
Router
Navigation.Provider
Location.Provider
DataRoutes
RenderErrorBoundary
RenderedRoute
Route.Provider
RenderedRoute
Route.Provider
OrderDetailPage (0.5ms of 2.2ms)
Profiler (<0.1ms of 1.7ms)
div (Styled) (<0.1ms of 1.7ms)
div (Styled) (<0.1...
div (Styled) (<0.1...
div (Styled) ...
div (Styled)...
div (Styled)...
div (St...
div (St...
div (St...

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



Agenda

1. Cache 활용법
2. Font 최적화
3. CSS 최적화
4. 최적화는 아니지만…
    a. Lock Files
    b. TypeScript

Cache

1. 자주 사용하는 데이터를 저장하는 것
    a. 계속 불러오지 않고 이미 불러온 기록이 있으면 거기서 꺼내오는 것
2. 종류는 2가지
    a. Memory cache
    b. Disk cache
    c. 브라우저가 알아서 선택함
3. Cache-Control을 설정해주어야함

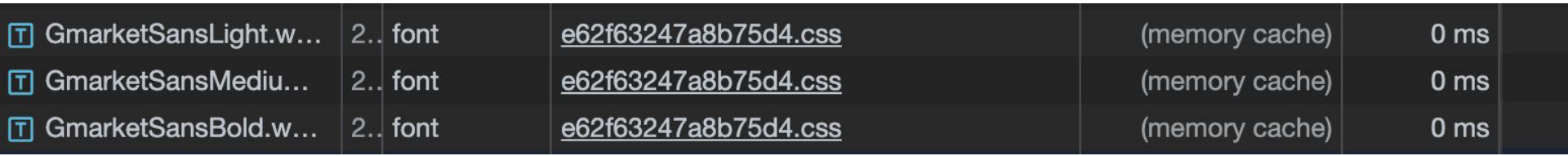
Cache

1. 자주 사용하는 데이터를 저장하는 것
    a. 계속 불러오지 않고 이미 불러온 기록이 있으면 거기서 꺼내오는 것
2. 종류는 2가지
    a. Memory cache
    b. Disk cache
    c. 브라우저가 알아서 선택함
        i. 사용빈도
        ii. 파일 크기
            1. 너무 크면 메모리에 저장할 수 없고 디스크로 가야함
3. Cache-Control을 설정해주어야함

Cache

GmarketSansBold.w…	2..	font	e62f63247a8b75d4.css	569 kB	64 ms
http://localhost:3000/_next/static/lGqLO2Jgojbbcb GK4KpN7/_buildManifest.js			e62f63247a8b75d4.css	435 kB	66 ms
GmarketSansMediu…	2..	font	e62f63247a8b75d4.css	487 kB	72 ms
site.webmanifest	2..	manifest	Other	567 B	8 ms
favicon.ico	2..	x-icon	Other	2.8 kB	10 ms
android-chrome-192…	4..	text/html	Other	1.3 kB	4 ms
terms	2..	document	Other	2.9 kB	2 ms
localhost	3..	document	Other	159 B	5 ms
e62f63247a8b75d4.css	2..	stylesheet	(index)	(disk cache)	2 ms
webpack-38cee4c0e…	2..	script	(index)	(disk cache)	1 ms
framework-a50f6e43…	2..	script	(index)	(disk cache)	9 ms
main-2cc097e2da96…	2..	script	(index)	(disk cache)	3 ms
_app-22d009cd32d6…	2..	script	(index)	(disk cache)	8 ms
53-4e8b1f37f7a71b3…	2..	script	(index)	(disk cache)	4 ms
index-8707e8d62281…	2..	script	(index)	(disk cache)	6 ms
_buildManifest.js	2..	script	(index)	(disk cache)	6 ms
_ssgManifest.js	2..	script	(index)	(disk cache)	2 ms
GmarketSansBold.w…	2..	font	e62f63247a8b75d4.css	(disk cache)	3 ms
GmarketSansLight.w…	2..	font	e62f63247a8b75d4.css	(disk cache)	2 ms
GmarketSansMediu…	2..	font	e62f63247a8b75d4.css	(disk cache)	3 ms

Cache



GmarketSansLight.w…	2..	font	e62f63247a8b75d4.css	(memory cache)	0 ms
GmarketSansMediu…	2..	font	e62f63247a8b75d4.css	(memory cache)	0 ms
GmarketSansBold.w…	2..	font	e62f63247a8b75d4.css	(memory cache)	0 ms

Font 최적화

1. WOFF2가 가장 용량이 작은 폰트
   a. https://transfonter.org/ 에서 폰트 포맷을 변경할 수 있음
2. https://github.com/fontsource/fontsource 으로 설치하는 방법

CSS 최적화


Reduce unused CSS
0.30s
Reduce unused rules from stylesheets and defer CSS not used for above-the-fold content to decrease bytes consumed by network activity. Learn how to reduce unused CSS. FCP LCP
URL
Transfer Size
Potential Savings
localhost 1st Party
52.6 KiB
41.1 KiB
…css/main.5a9c0adf.css (localhost)
52.6 KiB
41.1 KiB

PurgeCSS

1. https://purgecss.com
2. https://purgecss.com/CLI.html

CSS 최적화

Before

```
drwxr-xr-x  4 byeongjinkang  staff     128 Aug 10 19:40 .
drwxr-xr-x  4 byeongjinkang  staff     128 Aug 10 19:40 ..
-rw-r--r--  1 byeongjinkang  staff  444835 Aug 10 19:40 main.5a9c0adf.css
-rw-r--r--  1 byeongjinkang  staff  899513 Aug 10 19:40 main.5a9c0adf.css.map
```

After

```
drwxr-xr-x  4 byeongjinkang  staff     128 Aug 10 19:31 .
drwxr-xr-x  4 byeongjinkang  staff     128 Aug 10 19:31 ..
-rw-r--r--  1 byeongjinkang  staff  225970 Aug 10 19:31 main.5a9c0adf.css
-rw-r--r--  1 byeongjinkang  staff  899513 Aug 10 19:31 main.5a9c0adf.css.map
```

최적화는 아니지만 - Lock files

1. `package.json` vs `package-lock.json`
   a. `yarn.lock`
2. dependency를 정확하게 관리해야 여러 개발자들이 같은 환경에서 개발할 수 있음
3. 패키지는 하나만 사용해야함

최적화는 아니지만 - TypeScript

```typescript
interface Props {
  filter: {
    age: string;
    mbti: string;
    gender: string;
  };
  onChangeFilter: (e: React.ChangeEvent<HTMLSelectElement>) => void;
  onDeleteFilter: (name: string) => void;
}
```

1. 에러 표시 없이 정상작동하는 코드
2. 하지만 `filter`의 `key`만 `onChangeFilter`와 `onDeleteFilter`에서 사용되는 편이 더 안정적

최적화는 아니지만 - TypeScript

```tsx
const useFilterStatistics = (postId: number) => {
  const [filter, setFilter] = useState<Filter>({
    age: "",
    mbti: "",
    gender: "",
  });

  const [commentFilter, setCommentFilter] = useState<CommentFilter>({
    age: null,
    mbti: null,
    gender: null,
    sortBy: "ByTime",
  });
```

```tsx
interface CommentFilter {
  age: string;
  mbti: string;
  gender: string;
  sortBy: string;
}

You, 1 second ago | 1 author (You)
interface Filter {
  age: string;
  mbti: string;
  gender: string;
}
```

1. `key`가 겹치는 타입을 어떻게 선언할 것인가?
    a. Extends
    b. Pick

최적화는 아니지만 - TypeScript

```tsx
type ButtonProps = {
  children: JSX.Element;
  onClick?: (e: React.MouseEvent<HTMLButtonElement>) => void;
  info?: string;
  className?: string;
  infoPlace?: "top" | "bottom" | "left" | "right";
};

const CircleButton = ({
  children,
  onClick,
  info,
  className,
  infoPlace,
}: ButtonProps) => {
```

1. Optional prop은 default prop을 넣어주는 것이 좋음
   a. https://github.com/jsx-eslint/eslint-plugin-react/blob/master/docs/rules/require-default-props.md

[아하!모먼트] 이 회사에 지원한 이유가 무엇인가요?

1. 회사 규모에 따른 고민
   a. 초기 스타트업
   b. 중견기업
   c. 대기업
2. 개발문화에 따른 고민
3. 나는 여기서 무엇을 할 수 있는가?
4. https://f-lab.kr/blog/criteria-choosing-company-for-developers