

16. CSS Optimization

CSS 코드 최적화.



CSS Optimization

Remove unused CSS.

사용하지 않는 CSS 제거.

Eliminate render-blocking resources.

렌더 차단 리소스 제거.

“

Remove unused CSS



Remove unused CSS

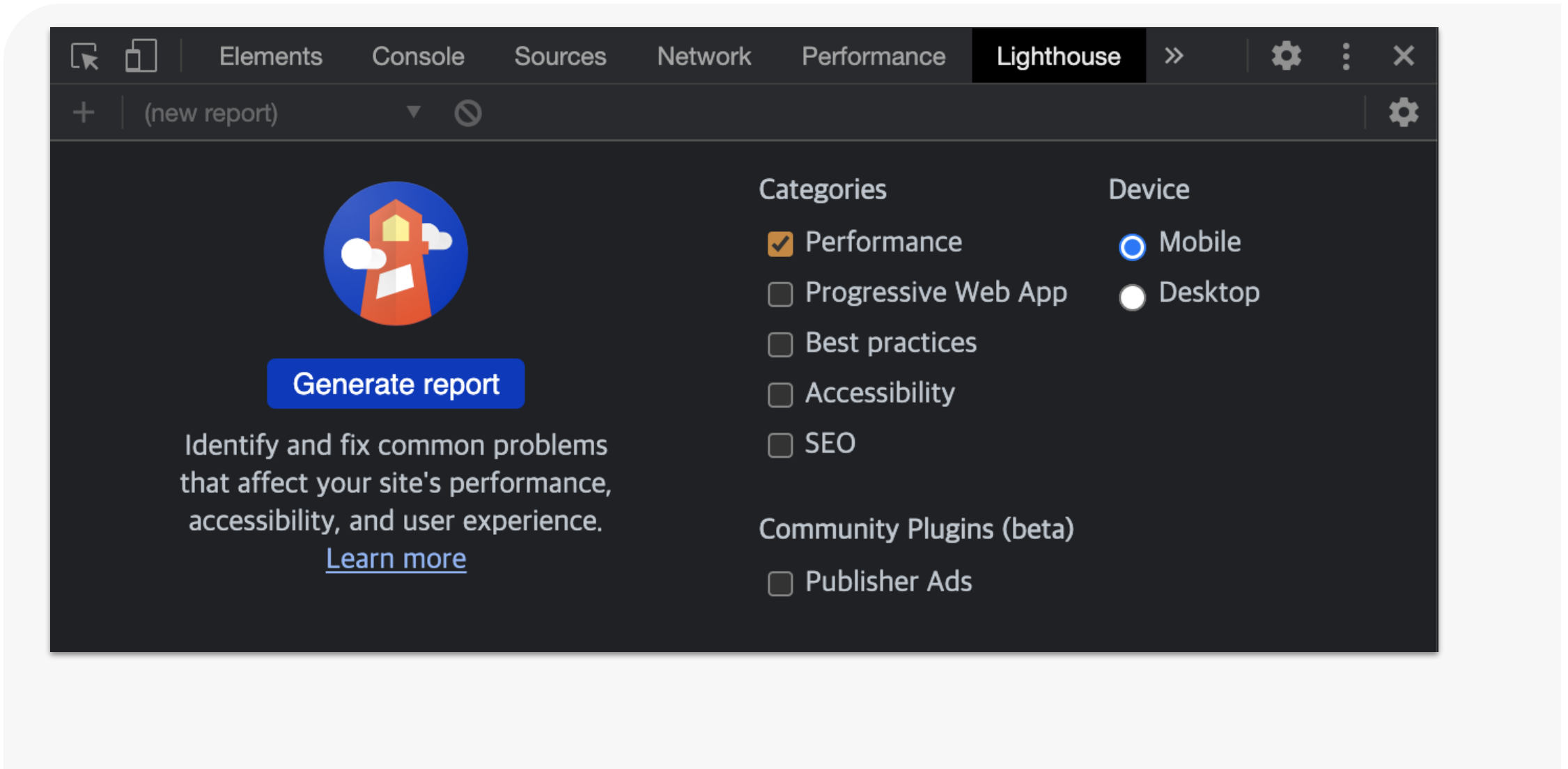
Unused CSS 왜 문제인가?

CSS는 페이지 렌더링을 차단하는 리소스. 브라우저가 스타일을 계산하는데 잠재적으로 더 많은 시간을 소비.

Remove unused CSS

구글 라이트하우스는
2KB 이상 미사용 CSS가 포함된 파일을 검출.

Google Lighthouse - Performance



Remove unused CSS



Remove unused CSS



1.95 s



Remove dead rules from stylesheets and defer the loading of CSS not used for above-the-fold content to reduce unnecessary bytes consumed by network activity. [Learn more.](#)

☒ Show 3rd-party resources (2)

URL

Transfer
SizePotential
Savings

...build/w.min.20210513c.css (mm.pstatic.net)

207.2 KiB

185.7 KiB

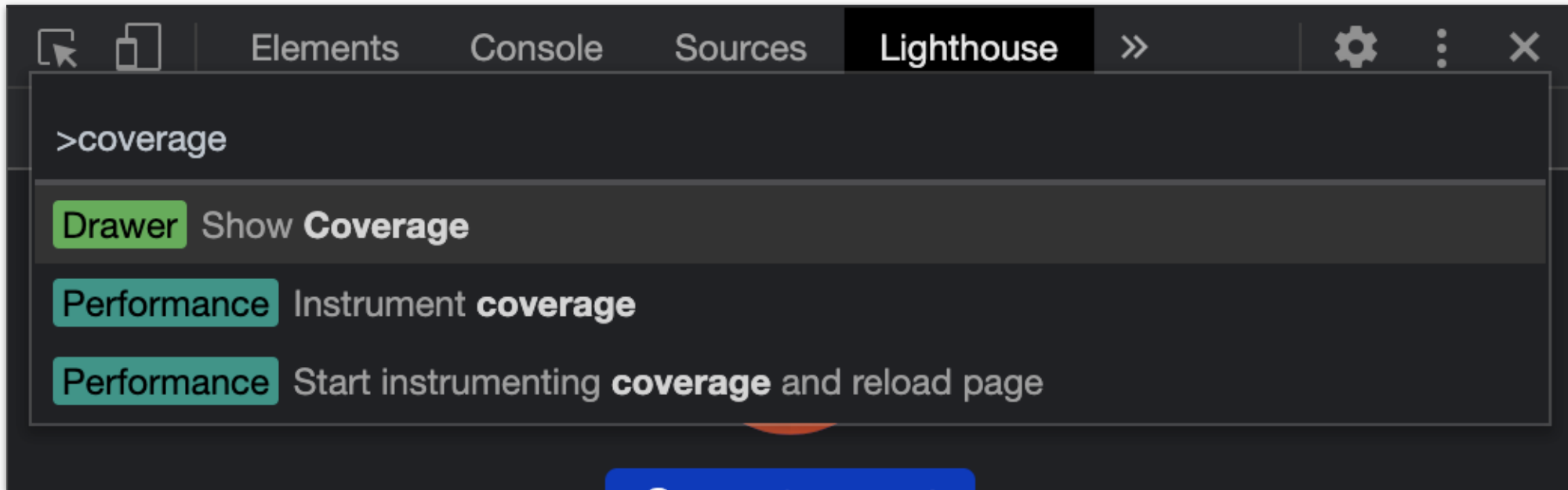
...build/w_dark.min.20210513b.css (mm.pstatic.net)

25.4 KiB

25.4 KiB

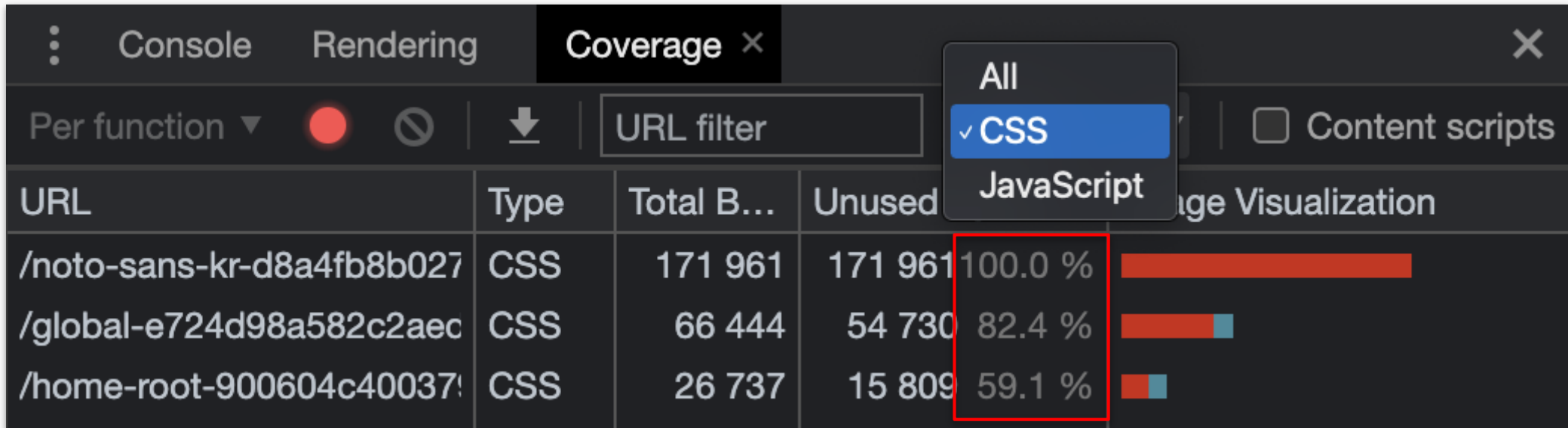
Coverage Tab

Cmd + Shift + P / Ctrl+Shift+P



Coverage Tab

Coverage 탭을 통해 unused CSS 확인



URL	Type	Total B...	Unused	Percentage	Visualization
/noto-sans-kr-d8a4fb8b027	CSS	171 961	171 961	100.0 %	<div></div>
/global-e724d98a582c2aec	CSS	66 444	54 730	82.4 %	<div></div>
/home-root-900604c40037	CSS	26 737	15 809	59.1 %	<div></div>

Check unused CSS

Unused CSS 코드의 실체



The screenshot shows a CSS editor window titled "home-root-90060....css:formatted". The code is as follows:

```
21  
22 @media (max-width: 639px) {  
23     .hero {  
24         margin-bottom: 16px;  
25     }  
26 }  
27  
28 .hero::before, .hero::after {  
29     content: '';  
30     position: absolute;  
31
```

The status bar at the bottom indicates "Line 19, Column 6" and "Coverage: 40.9 %".

“

Eliminate render-blocking resources



Render blocking resources

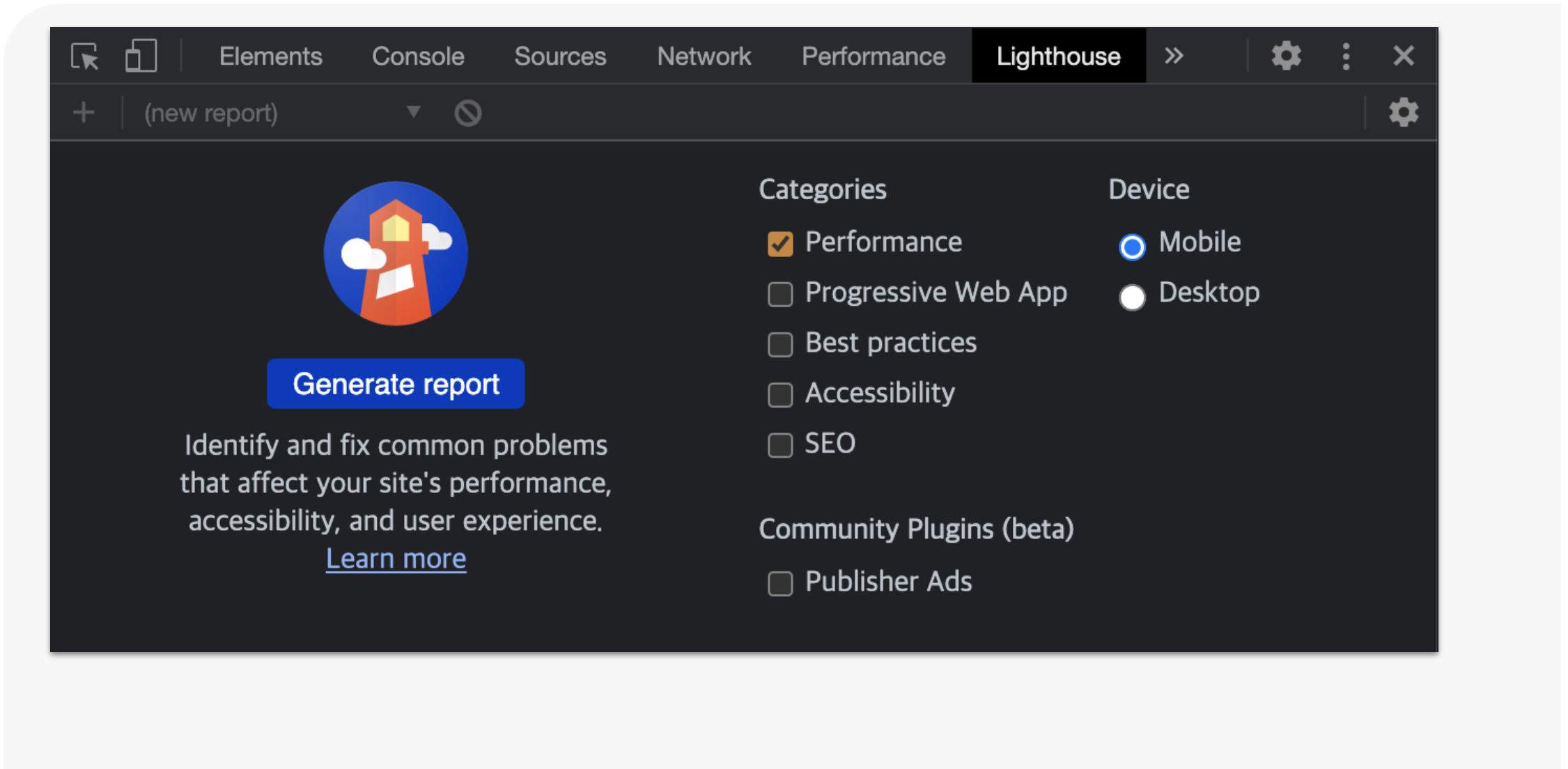
Render blocking 왜 문제인가?

브라우저가 외부 리소스를 다운로드하고 파싱하는 동안 페이지 콘텐츠를
를

파싱하거나 렌더링하지 않기 때문에 페이지 표시 속도 저하의 원인.

Unused CSS는 Render blocking을 가중하는 요인.

Google Lighthouse - Performance



Render blocking resources

▲ Eliminate render-blocking resources

1.53 s ^

Resources are blocking the first paint of your page. Consider delivering critical JS/CSS inline and deferring all non-critical JS/styles. [Learn more](#).

☒ Show 3rd-party resources (1)

URL	Transfer Size	Potential Savings
...build/w.min.20210513c.css (mm.pstatic.net)	207.2 KiB	1,810 ms

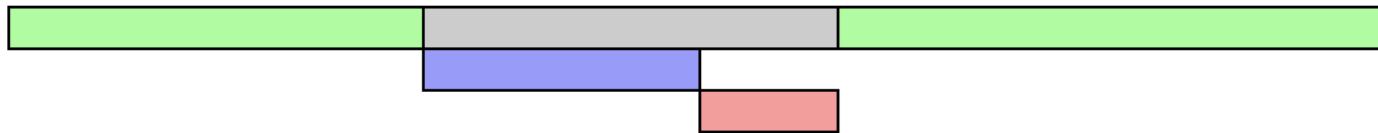
Render blocking resources

렌더 블로킹 리소스 표시 조건:

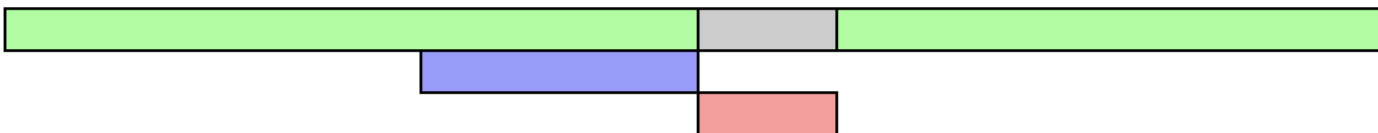
- defer, async 속성이 없는 <head> 요소의 <script> 태그.
- media 속성과 값이 없는 <link rel="stylesheet"> 태그.

Render blocking <script>

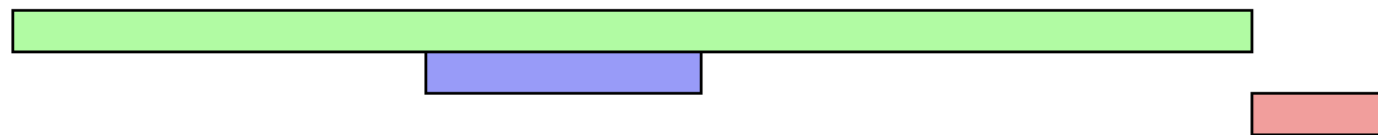
No async. No defer.



async: 병렬 다운로드, 즉시 실행



defer: 병렬 다운로드, 지연 실행



Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution

Article - <https://t.ly/KKLr>

Render blocking <script>

// 병렬 다운로드, 즉시 실행

```
<script async src="script.js"></script>
```

// 병렬 다운로드, 지연 실행

```
<script defer src="script.js"></script>
```

Render blocking <script>

1. 필수 스크립트는 html에 <script> 형식으로 작성.
2. 기타 스크립트는 </body> 종료 태그 직전에 선언.
3. 마지막에 파싱해도 문제 없으면 defer 속성.
4. 가능한 빠른 시점에 실행 필요하면 async 속성.

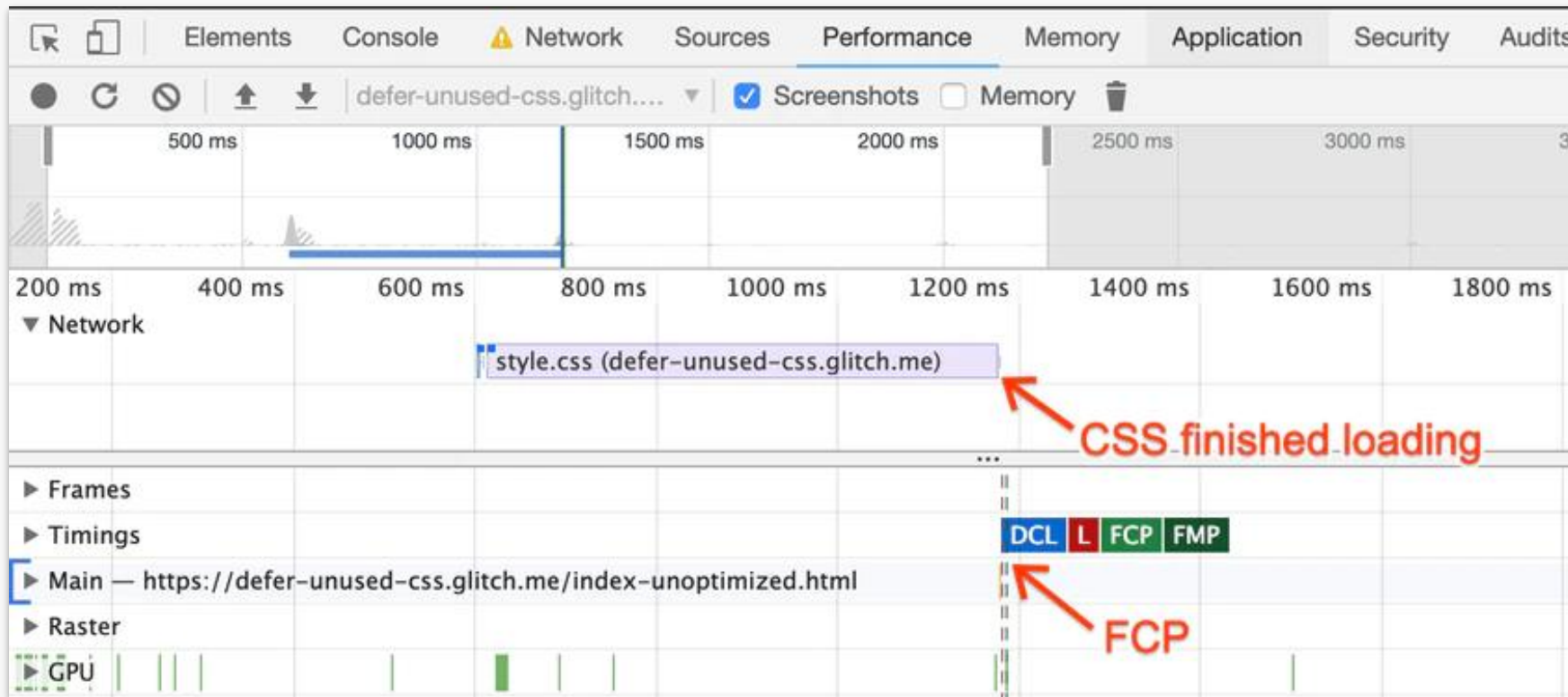
Render blocking <link rel="stylesheet">

media 속성이 없거나 값이 all이면 렌더 차단 리소스

```
<!-- Render blocking resource -->
<link href="style.css"      rel="stylesheet">
<link href="style.css"      rel="stylesheet" media="all">
<!-- Render blocking resource -->
<link href="portrait.css" rel="stylesheet" media="orientation:portrait">
<link href="print.css"    rel="stylesheet" media="print">
```

Render blocking <link rel="stylesheet">

CSS 파일이 렌더링을 차단하는 과정.



<https://t.ly/GiGe>
<https://is.gd/2jWuot>

Render blocking <link rel="stylesheet">

방법 1)

반응형 웹인 경우 해상도 구간별로 CSS 파일을 분리하고 media 속성으로 분리.

```
<link href="*.css" rel="stylesheet" media="(max-width:639px)">  
<link href="*.css" rel="stylesheet" media="(min-width:640px) and (max-width:960px)">  
<link href="*.css" rel="stylesheet" media="(min-width:961px)">
```

Render blocking <link rel="stylesheet">

방법 2)

1. 필수 스타일은 페이지 <head>에 <style> 형식으로 작성.
2. 지연 스타일은 <link rel="preload"> 속성으로 병렬 로딩 후 지연 적용.

Render blocking <link rel="stylesheet">

방법 2)

필수 스타일 임베딩, 지연 스타일 병렬 로딩 후 지연 적용.

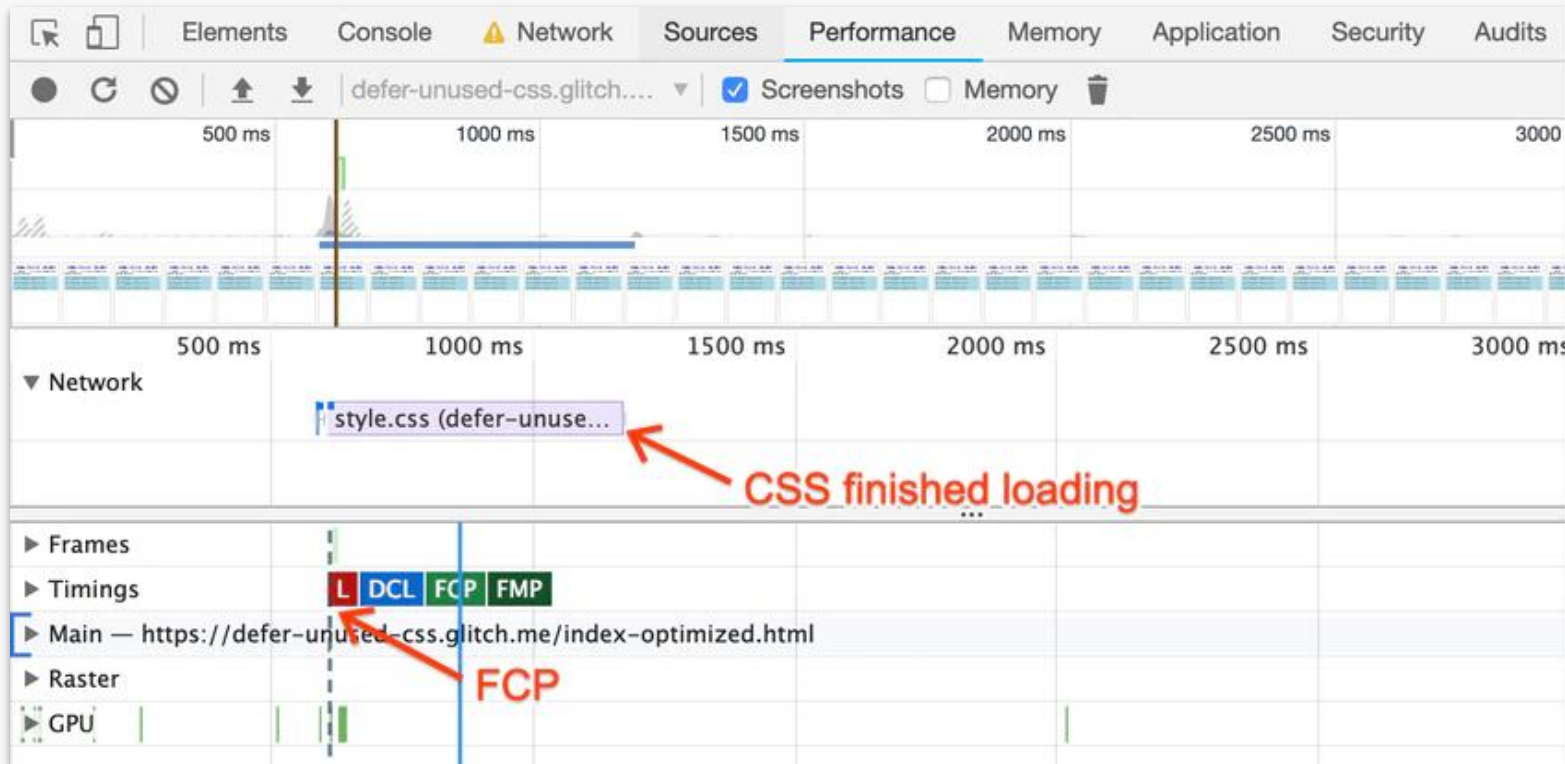
```
<style>
  /* 필수 스타일 여기 */
</style>
<link rel="preload" as="style" href="x.css"
onload="this.onload=null;this.rel='stylesheet'">
```

this.onload=null 할당 이유:

rel 속성을 변경할 때 일부 브라우저가 다시 onload 실행하는 것을 방어하는 코드.

Render blocking CSS

방법 2) 외부 스타일 파일이 렌더링(FCP)을 차단하지 않음.



Article - <https://t.ly/GiGe>

Demo - <https://is.gd/HuwPS2>

“

SUMMARY



“ Summary

1. 웹 브라우저는 외부 JS, CSS 파일을 로딩 하고 파싱 하는 동안 렌더링 차단 상태를 유지한다.
2. 사용하지 않는 JS, CSS 제거.
3. 필수 코드는 페이지에 `<style>...</style>`, `<script>...</script>` 작성하기.
4. 필수 아닌 JS는 `</body>` 종료 직전 위치를 고려. `defer`, `async` 속성을 사용.
5. 필수 아닌 CSS는 병렬 로딩(`preload`)하고 지연 적용(`onload`)하기.

“

실습 과제



실습 과제

1. <https://github.com/naradesign/css> 저장소를 포크하세요.
2. defer-css.css 파일에서 Unused CSS 코드를 찾아 제거하세요.
3. defer-css-unoptimized.html 파일에서 필수 CSS 코드를 <head> 내부에 추가하세요.
4. defer-css-unoptimized.html 파일에서 렌더 블로킹 CSS를 병렬 로딩 (preload)하고 지연 적용(onload)하세요.
5. <https://github.com/naradesign/css> 저장소에 Pull Request를 보내주세요.

⚠ 참고: 보내주신 Pull Request는 병합하지 않습니다.

Thank you !

