4주차 로드과정 최적화 리서치 제출

학습자

박민규 (TDD/4기)

**1. 과제 개요**

본 과제는 4주차 학습 내용을 바탕으로, 스파르타코딩클럽 사이트의 로드 과정에서 최적화가 필요한 부분을 리서치하는 것입니다.

학습자들은 사이트의 로드 성능을 분석하고, 개선이 필요한 부분을 찾아내어 최적화 방법을 제안합니다.

**2. 과제 내용**

**1. 사이트 로드 성능 분석**

스파르타코딩클럽 사이트를 로드하여 성능을 분석합니다. LightHouse 성능 분석 도구를 사용하여 로드 시간을 측정하고, 주요 성능 지표를 확인합니다. 주요 성능 지표는 LCP, INP, CLS, FCP, TTFB, 총 5가지로 분석하여 확인합니다. 평가자료로 활용할 자료는 사용자의 경험을 분석하여 확인한 코어 웹 바이탈 평가자료와, 성능문제 진단자료를 활용하여 분석합니다.

처음으로 확인할 자료는 코어 웹 바이탈 평가자료입니다. 해당 데이터를 PC와 모바일에서 확인한 결과를 각각 분석하여 성능 분석을 진행합니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

그림 1. PC에서 확인한 스파르타코딩클럽 홈페이지의 코어 웹 바이탈 평가자료

그림 1은 PC에서 확인한 스파르타코딩클럽 홈페이지의 코어 웹 바이탈 평가자료입니다. 지표에 따르면, LCP, INP, FCP, TTFB 모두 최적화되어있는 상태지만, CLS 성능 지표는 0.35초로 성능 개선이 필요한 것으로 분석할 수 있습니다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

그림 2. 모바일에서 확인한 스파르타코딩클럽 홈페이지의 코어 웹 바이탈 평가자료

그림 2은 모바일에서 확인한 스파르타코딩클럽 홈페이지의 코어 웹 바이탈 평가자료입니다. 지표에 따르면, 마찬가지로 LCP, INP, FCP, TTFB 모두 최적화 되어있는 상태지만, CLS 성능 지표는 0.67초로 성능 개선이 필요한 것으로 분석할 수 있습니다. 따라서, 코어 웹 바이탈 평가자료를 분석하는 경우, 기기에 상관없이 CLS 성능 지표를 개선하는 방향으로 개선해야 한다고 판단할 수 있습니다.

다음으로 확인할 평가자료는 성능문제 진단자료입니다. 해당 데이터를 PC와 모바일에서 각각 확인한 결과를 각각 분석하여 성능 분석을 진행합니다.

텍스트, 스크린샷, 웹 페이지, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

그림 3. PC에서 확인한 스파르타코딩클럽 홈페이지의 성능문제 진단자료

그림 3은 PC에서 확인한 스파르타코딩클럽 홈페이지의 성능문제 진단자료입니다. 지표에 따르면, 성능지표가 16점으로 매우 낮은 점을 확인할 수 있습니다. 측정항목은 FCP, LCP, TBT, CLS, SI를 확인하며, 각각의 점수가 1.3s, 4.4s, 1.12s, 0.497s, 3.8s임을 알 수 있습니다. 이는 모든 수치를 개선해야 한다고 판단할 수 있습니다.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

그림 4. 모바일에서 확인한 스파르타코딩클럽 홈페이지의 성능문제 진단자료

그림 4은 모바일에서 확인한 스파르타코딩클럽 홈페이지의 성능문제 진단자료입니다. 지표에 따르면, 성능지표가 19점으로 역시 매우 낮은 점을 확인할 수 있습니다. 측정항목은 FCP, LCP, TBT, CLS, SI를 확인하며, 각각의 점수가 8.0s, 19.4s, 1.5s, 0.215s, 13.9s임을 알 수 있습니다. 이는 모든 수치를 개선해야 한다고 판단할 수 있습니다. 따라서, 성능문제 진단자료를 통해 분석하는 경우, 기기에 상관없이 모든 성능 지표를 개선하는 방향으로 개선해야 한다고 판단할 수 있습니다.

**2. 최적화가 필요한 부분 식별**

성능 분석 결과를 바탕으로 로드 과정에서 최적화가 필요한 부분을 식별합니다. 사이트 로드 성능 분석을 진행하며 얻은 결론을 바탕으로 분석한 결과는 두 가지로 볼 수 있습니다. 첫 번째, 기기와 상관없이 성능개선이 필요하다. 두 번째, 모든 성능 지표를 개선해야 한다. 두 가지를 염두해서 최적화가 필요한 부분을 식별하자면 아래와 같습니다.텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

그림 5. 성능진단 평가지표를 통해 분석한 문제진단 자료

그림 5를 보면, 대규모 레이아웃 변경 피하기, 자바스크립트 실행 시간 단축, 콘텐츠가 포함된 최대 페인트 요소 등을 유의미하게 개선해야 함을 알 수 있습니다. 그 밖에 이미지 요소에 width, height 요소 명시되지 않음, document.write() 피하기, 레거시 JS 제공 등에 대한 문제를 해결해야 함을 알 수 있습니다.

**3. 최적화 방법 제안**

식별된 문제를 해결하기 위한 최적화 방법을 제안하고, 각 최적화 방법이 어떻게 성능을 개선할 수 있는지 설명합니다. 이를 위해 이미지 압축, JS코드 최적화, 코드 스플리팅, 캐싱 등의 방법을 제안할 수 있습니다.

이미지에 대한 작업을 진행하는 것이 가장 중요하다고 판단합니다. png 확장자와 같은 이미지 파일과 다르게, webp나 avif 확장자로 변경하여 이미지 파일을 압축하는 것으로, LCP, CLS, SI 수치를 개선할 수 있습니다. 또한, 같은 이유로 width와 height 요소를 추가하여 페이지를 구축하고, 적절한 이미지 지연로딩을 도입하여 UX를 개선하는 데에도 큰 도움이 될 수 있습니다.

다음으로, JS코드 최적화를 도입할 수 있습니다. 사용하지 않는 js 파일을 빌드하지 않고 배포하며, 코드에 polyfill을 적용하여 최신 브라우저에서 최적화된 js코드가 사용될 수 있도록 유도함으로써 페이지 로드에 있어서 부담을 줄일 수 있습니다. 또한, 병렬로 처리가능한 api 호출에 대해 비동기 호출을 통한 최적화를 진행하는 방법도 존재합니다. 이를 통해 FCP, TBT, SI 수치를 개선할 수 있습니다.

또한 코드 스플리팅을 통해 배포받는 파일의 부담을 줄일 수 있습니다. 페이지를 구성하는 가장 기본적인 부분을 우선적으로 보내, 사용자가 페이지를 먼저 확인할 수 있도록 유도한다면, FCP 수치를 개선할 수 있습니다. 이는 CDN을 전략적으로 이용하여 배포파일을 보낼 수 있다면 페이지 성능수치를 더욱 개선할 수 있습니다.

마지막으로 캐싱을 이용하여 재방문 시 페이지 수치 개선을 기대할 수 있습니다. 사진 데이터를 포함하여 자주 사용하는 파일을 브라우저나 메모리 단에 캐싱하여 사용한다면, 재방문 시 네트워크가 요구받는 것이 줄어들기 때문에 그에 따른 성능수치를 개선할 수 있습니다.

**4. 결과 및 요약**

해당 보고서로부터 스파르타코딩 홈페이지의 성능 분석 결과, 최적화가 필요한 부분, 제안된 최적화 방법에 대해 분석했습니다. 스파르타코딩 홈페이지의 전체 성능지표는 코어 웹 바이탈 성능지표에서 LCP, INP, FCP, TTFB, CLS 값을 분석하였고, 그 중 CLS 값의 성능개선이 필요함을 확인할 수 있었습니다. 또한 성능문제 분석지표에서 FCP, LCP, TBT, CLS, SI 값을 분석하였고, 그 중 모든 값의 성능개선이 필요함을 확인할 수 있었습니다. 성능지표에 대한 개선사항으로 대규모 레이아웃 변경 피하기, 자바스크립트 실행 시간 단축, 콘텐츠가 포함된 최대 페인트 요소 등을 유의미하게 개선해야 함을 확인할 수 있었습니다. 이를 해결하기 위해서는 이미지 압축, JS코드 최적화, 코드 스플리팅, 캐싱의 방법을 통해 성능지표를 개선할 수 있음을 확인할 수 있었습니다.

**5. 참고문헌**

1. PageSpeed Insights(2025). Google 2025.02.26 검색, <https://pagespeed.web.dev/analysis/https-spartacodingclub-kr/o9qijxxfbl?form_factor=desktop&category=performance&category=accessibility&category=best-practices&category=seo&hl=ko&utm_source=lh-chrome-ext()>