SKILLS

Programming Languages
TypeScript, JavaScript, HTML, CSS

Frameworks & Libraries
Next.js, React, React-native, Threejs

Styles
Emotion, SCSS, Tailwind CSS

State Management
Zustand, React-query, Redux, Recoil

SCHOOL

Brown International School
졸업, 고등학교
2018.10 ~ 2020.03

MILITARY

병장만기전역
2021.11 ~ 2023.05

ACADEMY

[구름 x 인프런] 자바 스프링 & 리액트 풀스택 개발자 성장 과정 3회차
수료
2023.10 ~ 2024.04

LANGUAGES

영어
공식 문서 바로 확인 가능,
영문 작성 / 스피킹 능숙함

CONTACT

Email
hiyunseok347@gmail.com

Portfolio
https://yunseokchoi.vercel.app

GitHub
https://github.com/HermannChoi

Velog
https://velog.io/@hiyunseok347/posts

안녕하세요, 디테일 바라기 프론트엔드 개발자, 최윤석입니다.

UI/UX를 중시하며 웹앱의 일체감과 사용자 만족을 목표로 하는 개발자입니다.
디테일한 디자인과 애니메이션으로 성능에 지장을 주지 않으면서도 매끄러운 사용자 경험을 구현합니다. 최근에는 반응형 포트폴리오 웹사이트를 만들어서 코딩능력을 향상시켰습니다.

포트폴리오 : https://yunseokchoi.vercel.app

PROJECTS

Palette* 2024. 04

개발자를 위한 채용 및 프로젝트/ 이력서 작성 서비스

기술스택 : Next.js, TypeScript, Emotion, Zustand, React-query

개발인원 : 6명(Front-End 3명, Back-End 3명)

담당역할 : UI/UX, 홈페이지, 로그인/회원가입, 이력서/포트폴리오 템플릿, 평판 작성

성능향상 : rest api에서 받아오는 response값의 id및 데이터를 전역상태관리에 담아서 같은 페이지를 들어갔을 시에 통신을 막아 서버부하 및 페이지 응답속도를 4배 개선시켰습니다.

트러블슈팅 :

• 어드민과 일반 이용자 로그인 후 동작 차이

로그인 후 API 응답을 받아서 role의 값을 쿠키에 저장한 다음, 그 값이 admin이면 어드민 페이지로 라우팅되게 구현하려고 했지만 실패했다. 자바스크립트의 비동기적 특성을 알아차리지 못하고 순서대로 진행된다는 안일한 생각을 알아차리고, async/await 함수 안에서 await의 response값을 사용하는 식으로 코드를 변경하고 문제를 해결할 수 있었다.

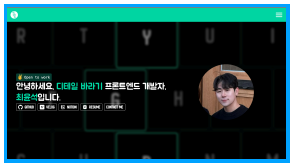
• 섹션 추가할 때 인풋의 변화

map()함수로 인풋을 렌더링하는 작업에서 버튼을 눌러 추가하고 제거하는 동작을 구현하는 과정에서 문제가 발생했다. 어떤 순서의 제거버튼을 누르던, 가장 마지막 순서의 인풋이 제거되는 것이다. input에 상태 value를 넣지 않았기 때문에 가상돔 비교 후 리렌더링 될 때 해당 input에 있던 input DOM의 attribute를 그대로 가져가서 마지막 것만 사라지는 것처럼 보였던 것이어서 인풋에 value를 넣고 해결하였다.

• 사진의 유무에 따른 HTTP 통신 header

API 통신을 할 때 문제가 발생했는데 http 관련 문제였다. 헤더에 content-type을 multipart/form-data로 해주었는데 사진을 넣지 않았을 때 에러가 나는 것이다. 그래서 여러 가지 방법을 시도하다가 사진을 넣지 않았을 때는 조건 삼항 연산자를 이용해서 header의 content-type을 application/json으로 하고, 사진이 있을 때는 multipart/form-data로 통신하게 구현하였다.

My Portfolio v2 2024. 08



프론트엔드 개발자 최윤석을 소개하는 포트폴리오 사이트

기술스택 : Next.js, TypeScript, Emotion, Recoil

개발인원 : 개인 프로젝트

성능향상 : 받아오는 이미지에 lazyLoading과 sharp를 사용 및 처음 렌더링되는 화면의 애니메이션 최소화를 통해 LCP를 줄여 페이지 로드시간을 4.6배 빠르게 만들고 lighthouse 87점의 성능에서 100점까지 올렸습니다.

트러블슈팅 :

• 이미지의 동적경로

포트폴리오의 테크스택에 맵을 돌려서 이미지를 넣으려고 경로를 사용하는데 src에 import를 해서 변수넣는 방식은 잘 되었지만 백틱으로 동적경로를 넣으니까 이미지가 로드되지 않았다. require()함수를 쓰지 않았기 때문인데, src에 백틱을 사용해서 동적으로 경로를 불러올 경우, 해당 경로를 string으로만 인식을 하기 때문에 경로를 찾지 못했던 것이다. require()함수로 백틱을 감싸주었고 해당 경로가 string에서 data-url로 인식이 됨으로써 이미지가 잘 로드될 수 있었다.

• 렌더링 차단 소스인 폰트 최적화

성능 최적화를 위해 lighthouse 점수 검사를 받고 폰트가 렌더링 소스 차단을 하고 있다는 걸 알고, 구글 검색 후 공부 후 css파일의 @import에서 html head부분에 link태그로 rel preload를 통해 렌더링 차단을 막을 수 있었다. 그리고 위 내용을 벨로그에 정리하여 미래에 같은 문제를 만났을 때 빠르게 해결할 수 있게 하였다.