김찬우

Email: hanrhfqkq@gmail.com

Github: https://github.com/chanooda

비즈니스도 잘하기 위해 노력합니다.

비즈니스 요구에 맞추어, 개발 시간과 품질을 조절합니다. 도메인 중심의 개발로 유지보수성을 높이고 오류 발생률을 낮춥니다.
페이지 이탈률, 구매율, FCP, LCP 등을 모니터링하며 개선합니다.

불가능한 것은 없다고 생각합니다.

문제를 해결할 때 최선의 방법을 찾아 해결하기 위해 노력하고, 안된다면 차선의 방법을 찾아 적용한 후에, 다시 최선의 방법으로 해결합니다.

성장하기 위해 노력합니다.

부족한 점을 보완해줄 수 있는 시스템 속에서 매 순간 주어지는 문제들을 해결하고, 팀원들의 말에 귀 기울이고, 의논한다면 어떠한 상황 속에서도 성장할 수 있다고 믿습니다.
팀원들과 함께 우리의 부족한 점을 파악하고, 개선 시키기 위해 의논하고, 우리들만의 시스템을 구축하고 착실히 실행하는 과정에서 성장합니다.

모든 걸 다 알 수는 없다고 생각합니다.

모든 문제에 대한 해결 법을 알려고 하기 보단, 나에게 주어진 문제의 해결법을 빠르게 찾아내어 적용하는 과정에서 성장합니다.
우리는 누구나 실수할 수 있고 부족할 수 있다는 점을 인정하고, 팀원들과 의논하며 문제를 해결하는 과정에서 성장합니다.

스마일드래곤

2023.06.14 ~ 재직중 / 웹 프론트엔드 개발자

- 사내 디자인 시스템 라이브러리 SDS v1, v2 개발 및 유지보수
- 마케팅24(서비스, 어드민) 개발 및 유지보수
- 뷰업(서비스, 어드민) 개발

마케팅24

온라인 마케팅 상품을 판매하고 제공하는 웹 사이트

Period 2023.12 ~ 2024.06

Member 프론트 4, 백엔드 5, PM 2, 디자이너 2

Link https://mkt24

Skills turborepo typescript next.js emotion @tanstack/react-query zustand react-hook-form

Point

- 월간 활성 사용자(MAU) 10,000명, 주간 활성 사용자(WAU) 2,600명을 기록하며 서비스 고착도(Sticky Rate) 26% 달성
- 누적 세션수 456,000여건, 참여 세션 수 371,000여건, 참여율 80%
- 상품 및 주문 등 주요 도메인 담당
- 메이저 릴리즈 5회
- SDI 도입으로 상품 추가 및 수정 시간 60분에서 30분으로 단축, 1년 6개월 동안 신규 상품 150여개 추가
- 상품 도메인 결함 발생률 40% 감소
- 메인, 상품 페이지 FCP, LCP 50% 개선
- 데이터 구조와 관련 모듈을 이용한 상품 관리 시스템 구축
- 상품 관리 공통 로직과 개별 상품 로직을 명확히 분리, 유지 보수성과 확장성 개선
- 상품과 관련된 각 로직을 독립적인 모듈로 구성함으로써, 코드 재사용성과 유지보수에서의 오류 발생률 개선

Detail

- server driven ui 적용

  상품의 빠른 추가와 수정을 위해 server driven ui를 도입하였고, 서비스 정식 출시 후 1년 6개월동안 기존 상품 30여개를 포함 150개 이상의 상품을 추가 및 관리하였습니다. 모든 상품 내의 옵션을 10분에서 30분내로 수정 가능하게 했습니다.

  마케팅24에서 관리하는 상품은 대부분 비슷한 폼 구조를 가지고 있었지만, 상품을 수정하거나 추가할 때마다 프론트엔드 코드를 수정해야 했습니다. 이로 인해 작업 속도가 느려졌고, 다양한 디자인에 빠르게 대응하는 데 어려움이 있었습니다. 주문 도메인은 사용자들이 가장 많이 이용하는 중요한 영역이었기 때문에, 효율적인 관리와 버그 발생률을 줄이는 것이 필요했습니다. 회사의 비즈니스 전략이 다양한 상품을 빠르게 추가하여 경쟁사와의 차별점을 갖는 것이었기 때문에, 빠르게 상품을 추가할 수 있어야 했습니다.

  일정 단위의 input을 컴포넌트로 나누고 variant를 지정했습니다. textField, radio, select 등 도메인과 연관되지 않은 순수한 input을 기준으로 나누었습니다. 또 sns정보를 불러오는 input, custom하게 댓글 리스트를 작성하는 textarea list 같이 도메인과 엮여있고, 많은 기능이 들어가 독립적으로 관리해주어야 하는 특수한 input을 기준으로 나누었습니다.

  각 input의 검증을 위해, schema 데이터를 추가했습니다. 프론트에서는 zod를 이용하여 문자열, 숫자, 범위, 복잡한 구조의 radioInput 등을 검증하는 schema를 상수로 관리하고, 해당 schema의 키값을 SDI 데이터에 variant를 기준으로 대응해 지정했습니다.

  서버에서 내려주는 SDI json 데이터를 클라이언트에서 적용하기 위해, 초기화 작업을 했습니다.

  react-hook-form의 formContext, zustand store를 통해 상품, 가격, sns 검증 값과 같은 client state에 각 컴포넌트가 독립적으로 접근하고, 기능할 수 있게 했습니다.

- server state 관리 및 LCP, FCP 개선

  기존에는 client side에서 외부 api의 데이터를 가져와 상품 페이지를 그려냈습니다. 문제는 상품에 대한 seo 처리를 할 수 없고, 페이지 컨텐츠의 첫 페인트가 느려지면서, 사용자 경험에 안좋

은 영향을 끼쳤습니다.

이 문제를 해결하기 위해서 server side에서 데이터를 가져오게 했습니다. 기존에는 client에서 tanstack-query를 이용해 캐시된 server state를 여러 컴포넌트에서 접근하는 구조였습니다. server side에서 불러온 데이터를 캐시해, client에서 재호출 하지 않고, 초기값으로 사용할 수 있게 했습니다. server side에서 사용하는 queryClient에 staleTime을 적용하고, prefetchQuery와 HydrationBoundery 컴포넌트를 이용했습니다.

- 결합도와 종속성을 낮추고, 단일 책임을 위한 전역 상태 관리

  기존에는 최상위 컨테이너에서 상품과 관련된 데이터를 하위 컴포넌트까지 props로 넘겨주었습니다. 그러다보니 props drilling이 심해졌고, 단일 책임을 위해 잘게 쪼개져 있던 컴포넌트와 custom hook에 불필요한 종속성이 발생했습니다.

  이 문제를 해결하기 위해 context와 zustand store를 사용했습니다. server에서 받아온 상품 관련 데이터를 createStore,context provider, useRef를 통해 초기화하고, useStore를 통해 접근할 수 있게 했습니다. 이로 인해 각 컴포넌트와 custom hook에서 props를 통해 상품 데이터를 관리할 필요가 없어지고, 불필요한 종속성을 가질 필요도 없게 되었습니다. 내부에서 state에 접근할 수 있기 때문에, 로직에 집중할 수 있고, 단일 책임을 지키기 수월해졌습니다

- FSD, custom hook 패턴을 이용한 관심사의 분리

  상품 도메인은 폼의 구현 및 검증, 가격 계산, 주문 진행 및 검증 등 여러 로직이 존재했습니다. 상품 페이지에는 여러 하위 도메인이 존재했습니다. 상품의 종류는 sns 상품과 post 상품이 존재했고, 각 상품들은 공통의 로직을 공유하지만 세부 구현이 달랐습니다. 코드 수정 시 각 도메인이 서로 영향을 끼치지 않도록 관심사의 분리가 필요했습니다.

  우선 각 도메인이 서로에게 영향을 미치지 않도록, FSD 구조를 이용해 product, sns, post 3개의 slice로 나누었습니다.
  각 기능들은 custom hook 패턴으로 구현했습니다. 기능의 구현은 세부적인 역할과 단일 책임을 강조하기 위해 3가지의 방식으로 분리했습니다. 비교적 저수준의 계산과 기능을 구현하는 helper 함수를 정의했습니다. hook과 연계해야 하는 세부적인 서비스 로직과 도메인 로직 구현을 위한 service 역할의 hook을 정의했습니다. 마지막으로 helper 함수와 service를 이용해 최종적인 도메인 로직을 구현하는 hook을 정의했습니다.

  예를 들어 상품을 초기화하는 기능을 구현할 때, product slice는 서버에서 받아온 SDI 데이터를 client에서 사용할 수 있게 mapping하고, 저장된 데이터를 관리합니다. sns, post slice는 form의 defaultvalues, schema, 가격 데이터를 초기화하는 로직을 각각 구현합니다. 마지막으로 product의 상품 초기화 로직을 최종적으로 담당하는 hook에서 상품의 종류에 맞는 도메인의 formSetting, 상품 데이터, 가격 데이터를 하위 store에 initial state로 전달합니다.

  이처럼 FSD와 custom hook 패턴을 통해 각 도메인의 로직을 독립적으로 관리함으로써 각 도메인에 영향범위에 대해서 걱정하지 않고 로직의 변경이나 확장, 유지보수에 집중할 수 있었습니다.

디자인 시스템 SDS

사내 프로젝트에 디자인 시스템을 빠르게 추가하고 적용하기 위한 디자인 시스템 확장 라이브러리

Period 2024.03 ~ 2024.06

Member 프론트 4, 디자이너 2

Link https://smiledragon-corp.github.io/design-system

Skills turborepo typescript emotion radix-ui vite

Point

- 메이저 릴리즈 2회
- 프로젝트 디자인 시스템 구성 기간 1달에서 1주로 단축
- 프로젝트 개별 컴포넌트 개발 기간 1일에서 10분으로 단축
- prop을 통한 스타일 확장 및 수정 기능으로 디자인 관련 컴포넌트 수정 3시간에서 30분으로 단축
- 환경과 시스템 구성 및 레이아웃 컴포넌트, textField, datePicker, checkbox, table, typography 컴포넌트 개발
- HTML 요소의 내장(native) 기능을 적극 활용하여 컴포넌트를 개발함으로써, 효율적이고 일관된 컴포넌트 동작 구현
- 테마 기반의 확장 기능 개발을 통해, 프로젝트 디자인 시스템 구성 방식을 개선

Detail

- Why

  자사의 다양한 서비스들은 각각 독립적으로 운영되고 있었기에, 하나의 공통 디자인 시스템을 사용하는 것보다 각 서비스별로 확장 가능한 베이스 디자인 시스템이 필요했습니다. 통일된 톤 앤 매너를 가진 서비스가 아니었기에, 각 서비스의 개별적인 요구 사항에 맞춰 디자인 시스템을 확장할 수 있도록 하기 위해 SDS 프로젝트를 시작했습니다.

- How
  - MUI 디자인 시스템 참고
    MUI 디자인 시스템 라이브러리를 참고하여, createTheme, themeComponent, defaultProps 기능을 통해 테마와 컴포넌트, 스타일의 수정 및 확장을 쉽게 할 수 있도록 했습니다. 이를 통해 각 서비스에서 필요한 디자인 시스템을, theme 객체의 확장만으로 쉽게 커스터마이즈할 수 있게 했습니다.
  - 네이티브 기능과 컴포넌트 확장
    컴포넌트를 단순히 제공하는 것에 그치지 않고, 기존 HTML 요소(Element)를 네이티브한 형태로 사용할 수 있도록 확장했습니다. 외부에서 주입된 이벤트 핸들러를 내부에서 한 번 더 오버라이드하여 기존의 Element 타입을 기반으로 확장 가능하게 만들었습니다.
    react-hook-form 라이브러리와의 연계성을 위해서 최대한 비제어 컴포넌트로 활용될 수 있게 했습니다.
  - 스타일 유틸리티와 테마 기반 스타일링
    테마 기반의 스타일 props와 styled 유틸 함수들을 도입하여, 컴포넌트 스타일을 외부에서 간편하게 주입하고 수정할 수 있게 했습니다. 이로 인해 코드의 유연성과 재사용성을 높였습니다.
    color, font, border, shadow 스타일을 위한 theme 기반의 code와 margin, padding, position 등 레이아웃 관련 스타일, CSSObject 형식으로 style을 정의할 수 있는 css 등 자주 사용되는 스타일에 관련된 값들을 props로 넘겨 확장성을 높혔습니다.
- Result
  - 디자인 시스템 적용
    신규 프로젝트에 SDS를 성공적으로 적용해, 개발 초기부터 일관된 디자인 시스템을 사용할 수 있었습니다. 이를 통해 개발 속도와 일관성이 크게 향상되었습니다.

- 기존 프로젝트의 디자인 시스템 수정 및 확장
  기존에 운영 중이던 두 개의 프로젝트에서도 SDS를 도입하여 디자인 시스템을 성공적으로 수정하고 확장했습니다. 이를 통해 기존 프로젝트에서의 유지보수와 기능 확장이 더욱 용이해졌습니다.
- 개발 효율성 및 일관성 강화
  프론트엔드 개발팀은 SDS를 통해 디자인 시스템을 중앙에서 관리하면서도 각 서비스별 요구에 맞게 확장할 수 있었으며, 코드의 일관성 유지와 재사용성이 높아져 개발 효율성이 크게 증가했습니다.

주식회사 라온스토리

2022.07.11 ~ 2023.06.10 / 프론트엔드 개발자

- 한국서부발전 계좌등록시스템, 한국서부발전 평가방시스템 개발 및 유지보수

한국서부발전 계좌등록시스템

온라인으로 계좌거래이체약정서를 발급 및 관리할 수 있는 웹 어플리케이션

Perid 2022.08 ~ 2023.03

Member 프론트 2, 백엔드 1, PM 1, 디자이너 1

Link https://ewha.iwest.co.kr/

Skills typescript react styled-component react-query recoil react-hook-form

Point

- 디자인 시스템 구성
- 계좌이체거래약정서 전자서명, 개인정보 마스킹 기능 구현
- recoil과 react query 도입으로 state관리 개선

자격증

- 정보처리기능사 - 2016.07.27
- 웹디자인기능사 - 2017.06.09
- 지도제작기능사 - 2017.09.29
- 컴퓨터그래픽기능사 - 2018.07.20
- 컴퓨터활용능력 1급 - 2020.05.01