황현규

이메일: joseph95501@gmail.com
휴대폰: 010-9049-9550
포트폴리오: https://01joseph-hwang10.github.io/
Github: https://github.com/01Joseph-Hwang10/

2년차 주니어 개발자 황현규입니다. 웹 애플리케이션 개발과 SaaS 창업에 관심이 많습니다.

창업 아이템을 내 손으로 직접 구현하기 위해 개발을 시작했으며, 다수의 해커톤 및 창업 활동에서 활약하면서 나의 아이디어를 직접 구현하는 과정에서 개발에 대한 흥미와 재미를 느껴 개발에 몰두하고 있습니다.

최소의 자원으로 최대의 효과를 추구하는 린 방법론에 크게 공감하며, 개발을 할때도 어떻게 하면 더 적은 코드로 더 많은 일을 할 수 있을지 고민합니다. 기능 구현에 불필요한 단계가 주기적으로 살피며, 무엇을 하는지에 대한 뚜렷한 목적을 가지고 업무를 진행합니다.

또한 지속가능한 개발을 지향하며, 명시적인 변수/함수 네이밍, 모듈화, OOP 등을 통해 간결하고 직관적, 유지보수가 용이한 코드를 작성하는데 힘써, 팀원이 처음 보는 코드라도 금방 이해하고 기여할 수 있게끔 노력합니다.

학력 사항

울산과학기술원 (2021.03 - 휴학 중)

- 컴퓨터공학과 학사과정 (주전공)
- 경영과학부 학사과정 (부전공)

경력 사항

트립빌더 (@tripbuilder)

- 2021.07 - 2021.12
- 개발팀 / Front-end Developer
- AI 기반 여행 계획 추천 모바일 앱 `TripBuilder` 개발 참여

랩이즈 (@labis)

- 2021.12 - 2022.11
- 개발팀 / Development Lead & Co-Founder
- 하드웨어 기반의 블록 코딩 교육 솔루션 `LearnQue` 기획 및 개발

쉐퍼드23 (@shepherd23)

- 2022.07 - 현재 재직중
- 개발팀 / Product Manager & Software Engineer

- E-Commerce 쇼핑몰을 위한 개인화 상품 추천 플러그인 PickHound 기획 및 개발

업무 경험

PickHound

=> Redux를 활용한 FLUX/MVC 기반의 React Component & Web Component (via StencilJS) 상태 연동

Frontend @shepherd23

- React와 StencilJS로 빌드된 Web Component의 상태 관리 방식이 다르기 때문에 각각의 경우에 맞는 상태 관리를 구현해야 했음.
- React, `@reduxjs/toolkit`의 `createSlice()` API를 활용해 `FLUX` 패턴을 기반으로 디자인 수정 페이지 구현
- `MVC` 패턴에 기반해 StencilJS Web Component의 style을 관리하는 `Banner Manager` 객체 작성
  - 스타일 객체를 `M`, `Redux Slice`를 `V`에 대응하게끔 구현하고 `C`에 해당하는 `Banner Manager` 객체를 작성.
  - `structedClone`을 활용한 새로운 Object Reference 생성으로 상태 업데이트
- `FLUX` 중 `Store`에 해당하는 `Redux Slice`를 `MVC` 중 `V`에 대응하여 두 컴포넌트의 상태를 연동.
- => 간결하고 직관적인, 유지보수에 용이한 상태 관리 코드 작성.
- => 디자인 패턴을 더 유연하게 바라보고 적재적소에 응용할 수 있는 시각을 함양.

> **팁**
>
> 아래 링크에서 위와 관련한 더 자세한 내용을 확인할 수 있습니다.
>
> https://01joseph-hwang10.github.io/posts/react-stenciljs-integration

=> Figma Admin Dashboard Kit에 기반한 디자인 시스템 구현

Frontend @shepherd23

- Figma Admin Dashboard Kit에 기반해 플러그인의 어드민 대시보드 개발을 위한 디자인 시스템 구현.
- 간결하고 직관적인 React 컴포넌트를 작성하기 위해 깊이 고민하고 노력함.
  - => `<Callout />` 컴포넌트를 통해 5분만에 서비스 사용자를 위한 새로운 메시지 배너 작성 후 배포.
  - => `<RoundedCard />`와 `<Container />`, `<HeadingWithLine />`을 활용해 간결한 코드로 UI의 영역을 구분.
  - => `colors` 상수 선언을 Figma Admin Dashboard Kit 디자인 시스템의 색상을 일관성 있게 관리 및 활용.

> **팁**
>
> 아래 링크에서 위와 관련한 더 자세한 내용을 확인할 수 있습니다.
>
> https://01joseph-hwang10.github.io/posts/admin-dashboard-ui-kit-implementation

=> Cafe24 API Auth Code 추출 자동화

Automation @shepherd23

- 실제 카페24 API를 활용해 개발을 진행하기 위해서는 카페24 API의 Authorization Code Flow 인증 과정을 거쳐야 함.
- 이에 작업을 위해 매번 브라우저를 열고 로그인 후 auth code를 추출하는 과정을 반복해야 하는 불편함이 존재했음.
- 이러한 비효율을 없애기 위해 Selenium을 활용해 로그인 및 auth code 추출 과정을 자동화함.
- => 개발 생산성 크게 향상. 더 많은 Iteration을 가능하게 함.

=> `Admin Dashboard`의 Auth Flow 테스트 효율화

Frontend Improvement @shepherd23

- Cafe24 API Auth Code 추출 자동화를 통해 개발 Iteration의 속도가 향상되었으나, auth code나 access token의 만료 등의 이유로 지속적으로 Auth Code 추출 스크립트를 실행해야 하는 불편함이 존재했음.
- 또한 Auth Flow가 복잡했기 때문에 어느 부분에서 에러가 발생했는지 파악하기 어려웠음.
- 이러한 문제를 해결하고자 `Admin Dashboard`의 Auth Flow를 테스트하기 위해 Cypress를 활용한 테스트 작성함.
- => Iteration 속도 향상. 개발 생산성 향상.

=> Code Split과 `<React.Suspense />`를 활용한 초기 로딩 UX 개선

Frontend Improvement @shepherd23

- MUI, D3등 다소 무거운 라이브러리 활용으로 `Admin Dashboard`의 번들 크기가 커지는 문제가 있었음.
- Code Split과 `React.Suspense`를 활용해 초기 로딩 UX 개선.
  - 별도로 분리된 번들이 다운로드 되는 동안 `fallback` 인자로 `<Loader fill />`를 전달해 로딩 화면을 렌더링.
  - => 초기 로딩 화면을 렌더링하는 동안 사용자가 멈춰있는 빈 화면을 보는 것을 방지.

See Also

- `AppRouter`: https://01joseph-hwang10.github.io/docs/career-description/pickhound#code-split과-reactsuspense-를-활용한-초기-로딩-ux-개선

=> StencilJS 도입을 통한 번들 크기 및 성능 최적화

Devops Optimization @shepherd23

- Cafe24로 제작된 쇼핑몰은 이미 많은 Javascript를 사용하며, 여기에 타사의 플러그인 또한 여럿 설치해 사용했음
  - => 큰 번들 사이즈는 엔드 유저의 UX 경험에 악영향을 미칠 수 있음을 인지함
- 기존의 `React` 대신 `StencilJS`를 도입하여 Web Components API의 이점을 활용하며, 번들 사이즈를 60% 이상 줄임
  - => 기능이 동일한 타사 추천 플러그인에 비해 60% 작은 번들 사이즈. (본사 35KB, 타사 85KB)

=> 다수의 IIFE 번들 빌드 프로세스 구성 및 CI/CD 자동화

Devops Automation @shepherd23

- 카페24에 배포해 쇼핑몰에 삽입되어 실행되는 스크립트는 브라우저에서 바로 실행될 수 있는 VanillaJS로써, 페이지마다 하나의 IIFE 번들이 존재해야 하는 제약조건으로 기존 SPA의 번들링 프로세스를 그대로 사용할 수 없는 상황이었음.
- StencilJS CLI와 Rollup, 배시 스크립트를 활용해 여러 개의 Entrypoint를 동시에 IIFE로 빌드하는 프로세스 구성함.
  - Webpack과 비교해, Scope Hosting 기능, 단순한 번들러 설정 및 CLI 활용성 등을 고려하여 Rollup을 도입함.
- Github Actions를 활용해 `Banner Manager`의 배포 자동화
- => 복잡한 빌드 및 배포 과정의 인적 자원 소비 최소화

=> 모노리포 구성, 패키지 버져닝 및 프로젝트 간 공통 모듈 공유

Automation Improvement @shepherd23

- Lerna, Python Module을 활용해 Monorepo 구성 (Recommendation API, Bandit Engine, Banner Manager)
- Lerna, `standard-version`을 활용한 패키지 버전 관리 (모든 프로젝트)
- 공유가 필요한 코드는 따로 패키지로 분리해 다른 리포지토리에서도 사용 가능하도록 구성 (Recommendation API, Banner Manager)
  - 공유 패키지 빌드 프로세스 구성 및 Github Actions, Google Artifact Registry를 활용해 자동 배포
  - => 비효율적인 복사 붙여넣기 방식의 코드 공유를 최소화.

=> 불필요한 코드 빌드 최소화

Optimization @shepherd23

- 대부분의 CI/CD 빌드 트리거가 Github Actions의 `push` 이벤트에 의해 발생함.
- 모노리포 상황에서 하나의 패키지에만 변경사항이 있는 경우에도 모든 패키지의 빌드가 발생하는 문제가 있었음.
- Lerna를 이용한 버저닝 관리와 함께 패키지 레지스트리에 배포되어 있는 패키지의 버전과 현재 레포지토리의 패키지 버전을 비교해 변경사항이 있는 경우에만 빌드가 발생하도록 구성함.
  - => 불필요한 빌드 최소화 및 CI/CD 자원 소비 최소화

=> Client Side 에러 트래킹을 위한 로깅 시스템 개발

Feature @shepherd23

- 서버에서 발생하는 에러와 달리 클라이언트 애플리케이션은 에러가 발생해도 팀 내에서는 인지하기 어려움.
- 고객의 오류 문의를 처리하기 위해서는 오류 당시의 자세한 상황을 알아야 하는데, 이를 고객의 설명에만 의존해 파악하기 어려움.
- 이에 Cloud Run에 POST request의 body를 `console.log` 를 통해 로깅하는 간단한 로그 서버와 이에 대한 클라이언트 라이브러리를 개발함.
  - 클라이언트는 `uuid` 를 활용해 고유의 에러 ID를 생성하고, 이를 로그 서버에 전송하며, 에러 ID를 에러 메시지와 함께 고객에게 전달.
  - 로그 서버는 Cloud Logging으로 로그를 스트림하고, 개발팀은 위에서 생성한 에러 ID를 통해 Cloud Logging에서 해당 에러를 검색.
- => 고객의 에러 문의에 발행된 에러 ID를 통해 즉각적인 문의 대응을 통한 고객 경험 향상.

=> 사내 내부 도큐멘테이션 배포를 위한 Private File Server 구축

Backend Devops @shepherd23

- 정적 웹페이지를 회사 내부 인원만 열람할 수 있도록 하는 Private File Server 구축.
- static-webpage-with-auth 로 도커라이즈 하여 여러 도큐멘테이션 페이지에 쉽게 적용할 수 있도록 만듬.

See Also

- static-webpage-with-auth: https://hub.docker.com/r/josephhwang02/static-webpage-with-auth

=> Google Sheets를 활용한 간이 어드민 대시보드 구축

Feature @shepherd23

- NestJS는 Django처럼 Admin Dashboard를 제공하지 않고, Thrid Party 라이브러리 (e.g. AdminJS) 도 Firestore 를 DB로 사용하는 경우를 고려한 라이브러리는 없었음.
- 이에 Google Sheets, Sheets API, Cloud Functions 를 활용해 사용자 데이터 시각화 및 쇼핑몰 관리 기능을 포함하는 간이 대시보드 구축함.
- => UI 구현 필요 없이 간단한 로직 구현만 필요했기에 빠른 대시보드 구현이 가능했음.
- => 엑셀에 기반한 대시보드 구현으로 비개발 인원도 쉽게 지표를 확인하고 쇼핑몰을 관리할 수 있었음.

=> 코드주석과 Notion을 연계한 사내 도큐멘테이션 작성 효율화

Devops Improvement @shepherd23

- 기존에는 비개발 인원이 개발 상황을 파악하기 위해 개발자에게 직접 문의하거나, 개발자가 이미 코드 주석으로 작성된 내용을 Notion에 다시 작성하는 등의 비효율이 존재했음.
- 이에 Swagger, Typedoc, Sphinx 등의 툴을 활용해 도큐멘테이션을 생성하고, 노션 페이지와 도큐멘테이션 페이지의 웹 링크를 연계하여 중복 작업을 최소화함.
- static-webpage-with-auth 를 개발해 `Google Cloud Run` 에 배포하여 사내 인원만 접근할 수 있도록 구성.

See Also

- static-webpage-with-auth: https://hub.docker.com/r/josephhwang02/static-webpage-with-auth
- 사내 내부 도큐멘테이션 배포를 위한 Private File Server 구축: https://01joseph-hwang10.github.io/docs/career-description/pickhound#사내-내부-도큐멘테이션-배포를-위한-private-file-server-구축

01joseph-hwang10.github.io

=> 고정된 이미지 가로세로비와 CSS 브라우저 호환성을 고려한 반응형 쇼케이스 이미지 구현

Frontend Feature

- `max-width: 100%; max-height: 100%;` 를 이용해 정해진 가로세로비의 이미지를 화면 너비에 따라 크기가 정해지는 쇼케이스 카드에 가장 크게 fit.
- 쇼케이스 카드의 빈 영역은 `before` Pseudo Selector와 `background-image`, `position`, `filter` 프로퍼티를 이용해 블러처리된 이미지로 Fill.
  - CSS 브라우저 호환성을 고려해 `backdrop-filter` 대신 `before` Pseudo Selector와 `filter` 프로퍼티를 이용해 구현. (Reference)
- => CSS 작동 방식에 대한 이해를 높이고 크로스 브라우징 이슈를 상시 인지하게 됨.
- => 원본 이미지와 블러된 이미지를 이용해 의도한 쇼케이스 레이아웃에 맞게 일관되고 깔끔한 UI/UX 제공

See Also

- Showcase Example: https://01joseph-hwang10.github.io/#Projects

=> 깔끔한 UI/UX를 위한 Docusaurus 소스 코드 Customize

Frontend Feature

- 도시의 풍경을 담은 "창문" 컨셉의 UI/UX를 제공하기 위해 항상 배경색이 존재하는 Docusaurus 네비게이션 바 코드 분석 및 Customize.
  - `@docusaurus/theme-classic`의 NavbarLayout 컴포넌트가 네비게이션 바의 컨테이너를 담당하고 있음을 확인.
  - `NavbarLayout` Swizzling 후 `@docusaurus/theme-common`의 `getNavbarHeight()` 함수 등을 Override.
  - 기존에 작성한 `useScroll()`, `useIsMobile()` 훅을 활용해 첫 화면에서 `NavbarLayout`의 배경색이 투명하게끔 구현.
- => 투명한 배경색의 네비게이션 바로 의도한 "창문" 컨셉의 시원한 UI/UX 제공
- => Facebook에서 제공하는 Docusaurus 소스 코드를 분석하면서 좋은 네이밍 컨벤션, 폴더 구조, 코드 스타일 등에 관한 깊은 인사이트를 얻을 수 있었음.

> **팁**
>
> 아래 링크에서 위와 관련한 더 자세한 내용을 확인할 수 있습니다.
>
> https://01joseph-hwang10.github.io/posts/portfolio-landing

=> CSS `position: sticky`와 JS `CustomEvent` API를 활용한 바로가기 툴바 구현

Frontend Feature

- 다소 까다로운 DOM 트리 내에서 거리가 먼 컴포넌트 간의 데이터 교환을 어떻게 하면 가장 간결하고 직관적으로 할 수 있을지를 고민함.
  - 기존의 방식인 `React.Context` API와 `Redux`는 불필요한 Boilerplate Code가 많고 상태 공유에 중간 컴포넌트가 필요함.
  - => JS `CustomEvent` API를 활용해 `subscribe`, `unsubscibe`, `dispatch` 이벤트 함수를 작성해 Higher Level Component 없이 컴포넌트간 직접적인 데이터 교환
  - => "데이터 교환"이라는 개념에 맞는 직관적인 코드 설계를 통해 코드의 가독성을 높이고 유지보수성도 크게 향상시킴.

> **팁**
>
> 아래 링크에서 위와 관련한 더 자세한 내용을 확인할 수 있습니다.
>
> https://01joseph-hwang10.github.io/posts/portfolio-shortcuts

=> Safari와 Chrome의 `max-*` 프로퍼티의 동작 차이 해결

Frontend Troubleshooting

- `max-width`와 `max-height` 프로퍼티가 Safari와 Chrome에서 동작하는 방식이 달라 브라우저 간의 아바타 컴포넌트의 표현이 달라지는 문제를 발견함.
  - Chrome은 childNodes의 `width`, `height` 프로퍼티를 우선시 -> 지정한 `max-width`, `max-height`까지 stretch.
  - Safari는 childNodes의 `width`, `height` 프로퍼티에 대해 조금 더 엄격함 -> 지정한 `max-width`, `max-height`까지 stretch하지 않음.
- `max-*` 프로퍼티와 함께 `width`, `height` 프로퍼티를 `100%`로 지정해 두 브라우저에서 동일한 표현을 보장.
- => 당연하다고 생각한 CSS 프로퍼티도 브라우저마다 동작 방식이 다를 수 있음을 인지하고, 항상 여러 브라우저를 이용해 테스트해야 함을 상기함.

LearnQue

=> Observer 패턴을 활용한 상태 관리

Frontend Improvement @labis

- Redux에 과도하게 의존적인 코드로 복잡한 블록코드 UI 개발에 비효율이 발생함.
- 이에 이벤트를 관리하는 `EventRegistry` 클래스를 구현하여, `EventRegistry.emit` 을 통해 이벤트를 발생시키고, `EventRegistry.on` 을 통해 이벤트를 구독하는 방식으로 상태 변경 사항을 관리하게끔 상태 관리 방식을 대거 수정함.
- => 불필요한 Redux 사용을 줄이고 코드 유지보수가 효율화

See Also

- Observer Pattern @ dbcav3: https://github.com/01Joseph-Hwang10/dbcav3#observer-pattern

=> OOP를 활용한 블록 코드의 블록 정의 개발

Frontend Feature @labis

- 각 블록정의는 `AbstractBlock`을 상속하고 `actionChain`이라는 메서드를 구현해야하며, 이 메서드는 상속과 참조를 통해 Redux Store로 연결되는 액션의 체인을 구성함.
- => `AbstractBlock`을 종점으로 상태 관리에 있어 단순함을 유지하며, OOP에 기반해 블록 정의의 확장성을 높임.

See Also

- Observer Pattern @ dbcav3: https://github.com/01Joseph-Hwang10/dbcav3#observer-pattern
- Block Definition Source Code: https://github.com/01Joseph-Hwang10/dbcav3/tree/main/src/modules/block-definitions

스킬셋

Typescript HTML5 CSS3 ES6 CommonJS Webpack Babel Rollup Vite Lerna
Python Numpy Pandas Scikit Learn Django Flask PyTest Selenium Sphinx React
NextJS Redux RTK Query TailwindCSS MUI Emotion Cypress Storybook StencilJS
D3 Figma NestJS Express Jest Swagger TypeORM Typedoc SQL Firestore
Cloud Computing Firebase Cloud Run Cloud Build Cloud Storage Cloud Logging
Cloud Compute Engine Docker Linux Github Git Github Slack Notion
Korean (Native) English (TOEFL iBT 98 @ 2023.02)

*최종 수정: **2023년 9월 14일**에*