ZARA웹페이지 개선

발표자: 고현우

2024-07-24

주제 선정 이유

- 기술적 이유:
- 기존 자라 웹사이트는 MPA(MultiPageApp) 방식으로 페이지 전환이 느림.
- React(빠른업데이트,렌더링)시 사용하면 성능향상, 사용성 개선가능.
- 디자인적 이유:
- 기존 자라 웹사이트는 UX가 굉장히 불편함.

기술스택

- 프로그램:
- GitHub DeskTop(Version 3.4.2)
- VisualStudioCode(Version 1.91.1)

- 사용언어:
- Html, Css, Sass, JavaScript, React

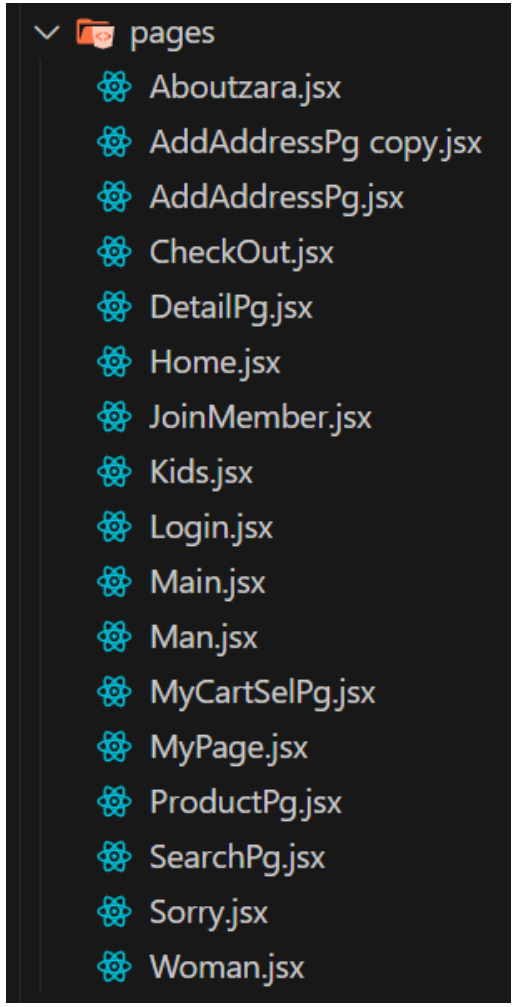
- 데이터베이스: 로컬스토리지 활용

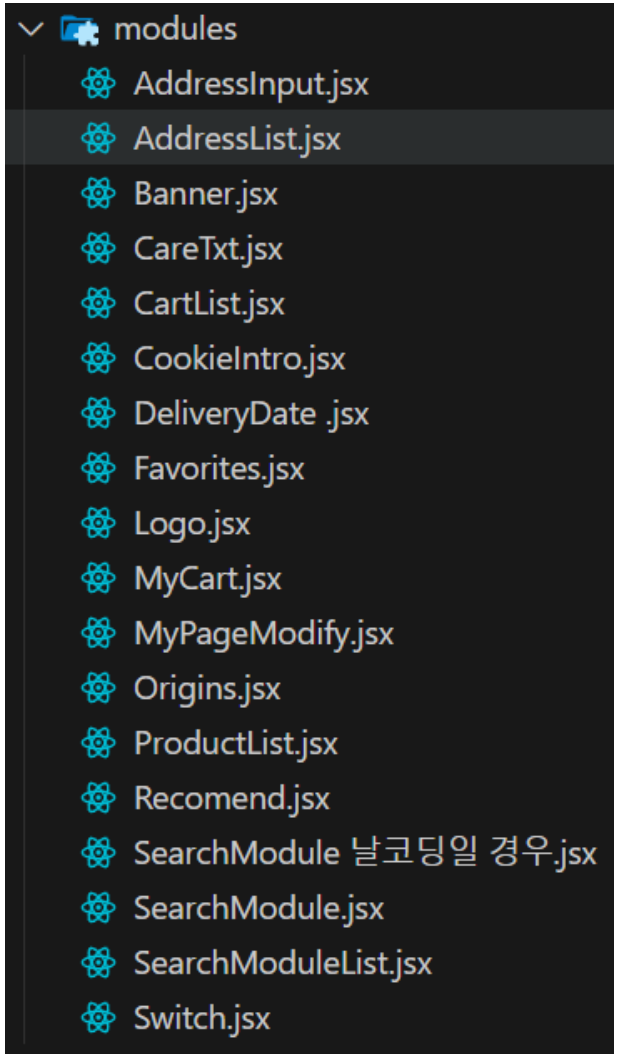
디자인 요소

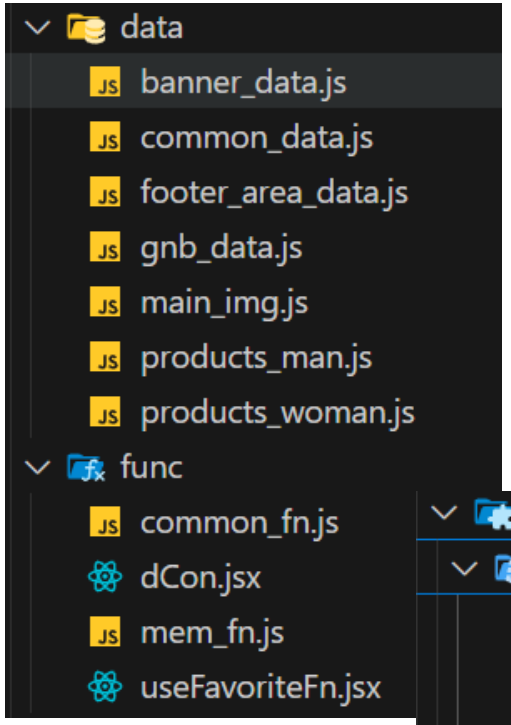
- 폰트:HelveticaRoman, HelveticaLT, Arial

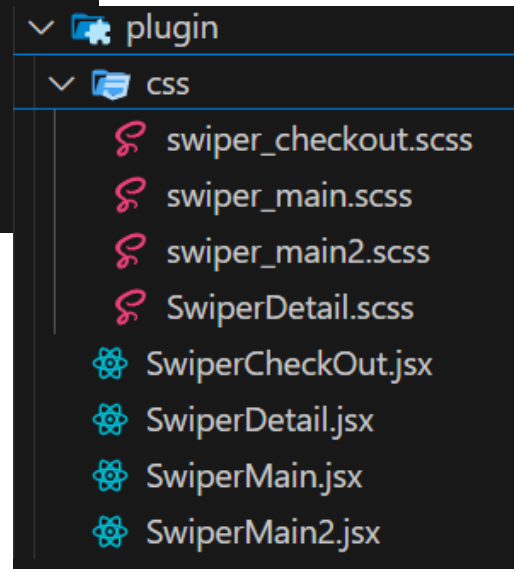
- 색상: #000, #FFF, #010101, #FDFDFD

구성요소


pages
Aboutzara.jsx
AddAddressPg copy.jsx
AddAddressPg.jsx
CheckOut.jsx
DetailPg.jsx
Home.jsx
JoinMember.jsx
Kids.jsx
Login.jsx
Main.jsx
Man.jsx
MyCartSelPg.jsx
MyPage.jsx
ProductPg.jsx
SearchPg.jsx
Sorry.jsx
Woman.jsx


modules
AddressInput.jsx
AddressList.jsx
Banner.jsx
CareTxt.jsx
CartList.jsx
CookieIntro.jsx
DeliveryDate .jsx
Favorites.jsx
Logo.jsx
MyCart.jsx
MyPageModify.jsx
Origins.jsx
ProductList.jsx
Recomend.jsx
SearchModule 날코딩일 경우.jsx
SearchModule.jsx
SearchModuleList.jsx
Switch.jsx


data
banner_data.js
common_data.js
footer_area_data.js
gnb_data.js
main_img.js
products_man.js
products_woman.js
func
common_fn.js
dCon.jsx
mem_fn.js
useFavoriteFn.jsx


plugin
css
swiper_checkout.scss
swiper_main.scss
swiper_main2.scss
SwiperDetail.scss
SwiperCheckOut.jsx
SwiperDetail.jsx
SwiperMain.jsx
SwiperMain2.jsx

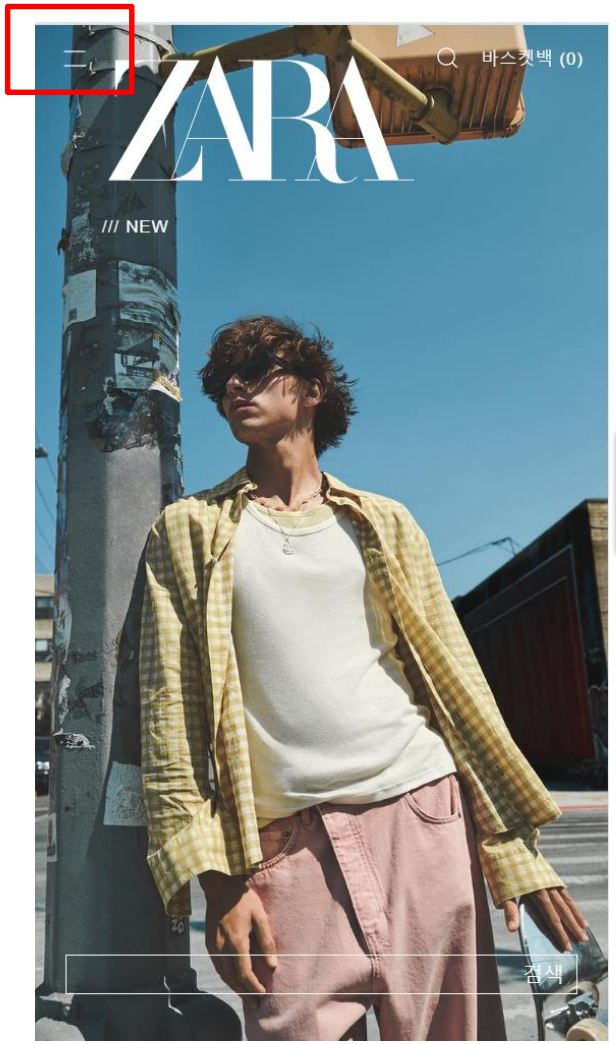
작업시 핵심 고려사항

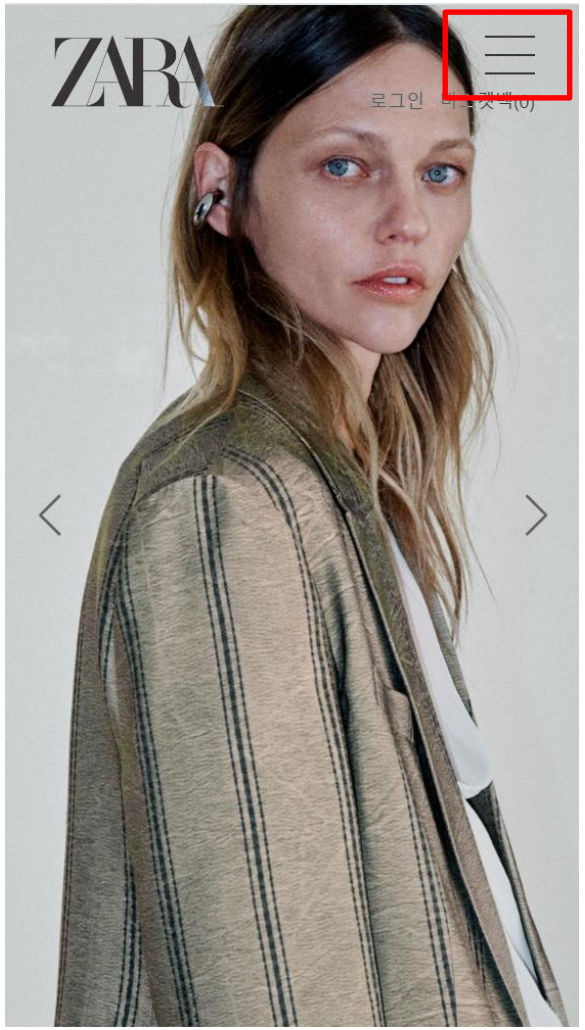
- ~~미디어 쿼리~~
- ~~유지 보수 편하게 하기~~

- 페이지 안정성

UI&UX개선

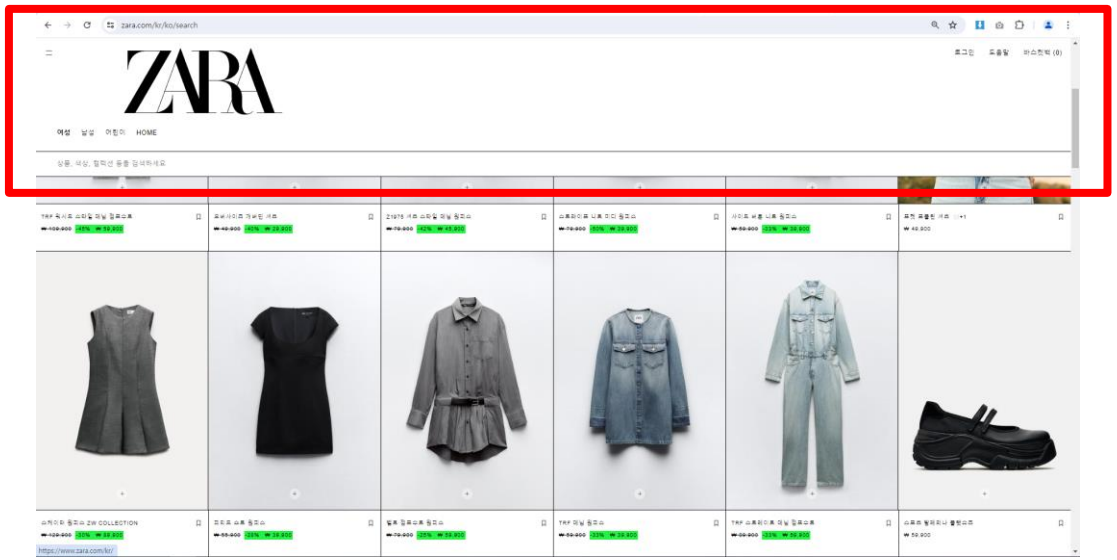
- 모바일페이지-
- 햄버거 버튼 크기
- 확대 및 위치 이동

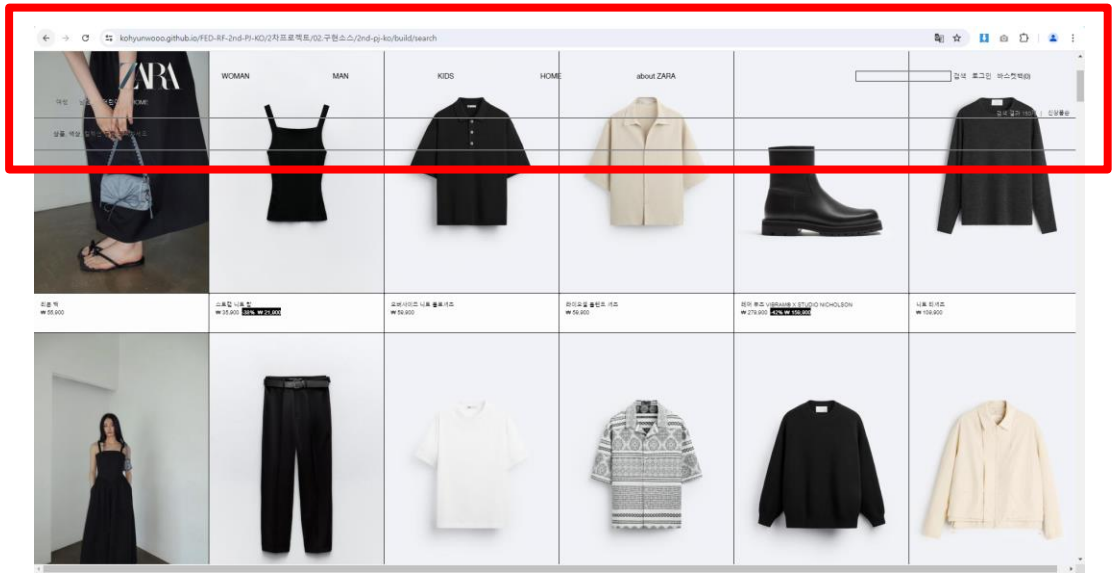




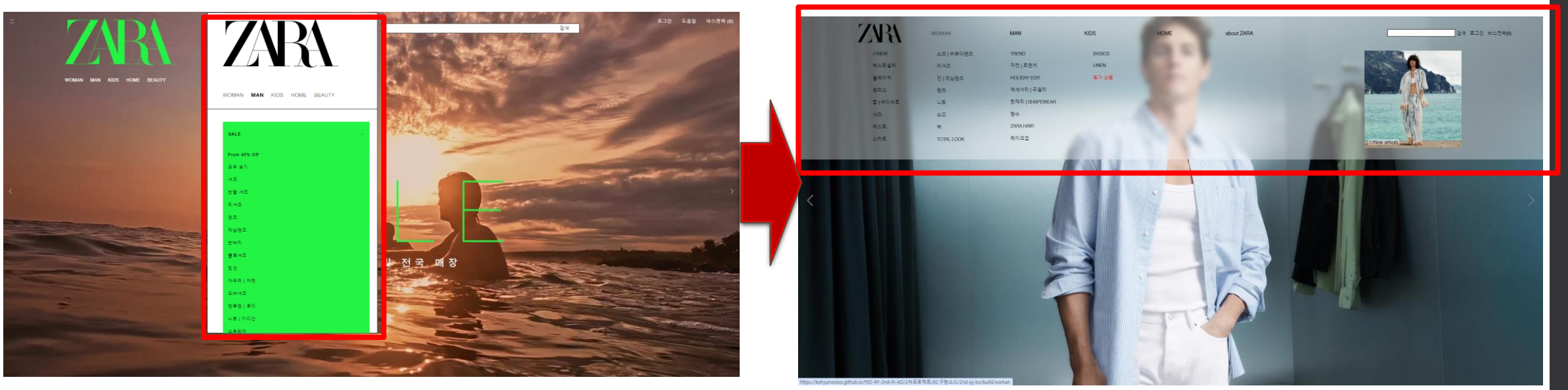
UI&UX개선

- 상품 화면 1/3가리는
- Position:sticky;
- search창 투명하게





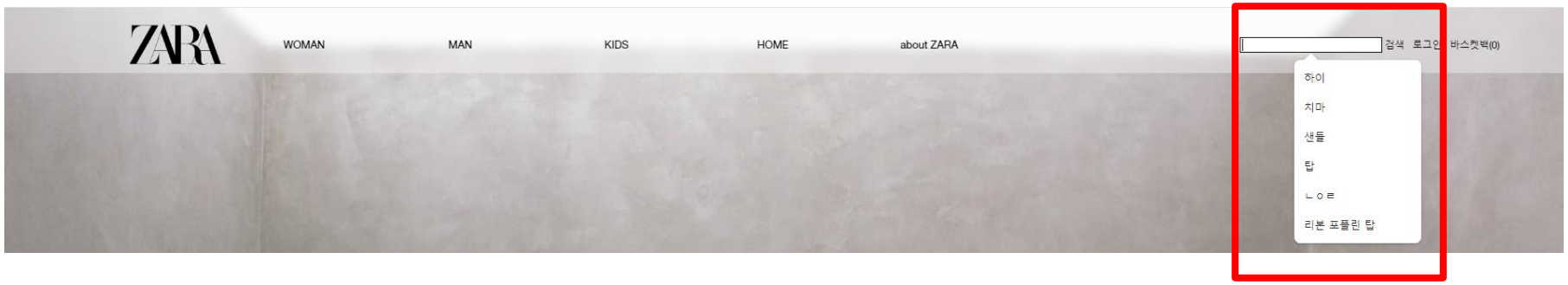
UI&UX개선



UI&UX개선

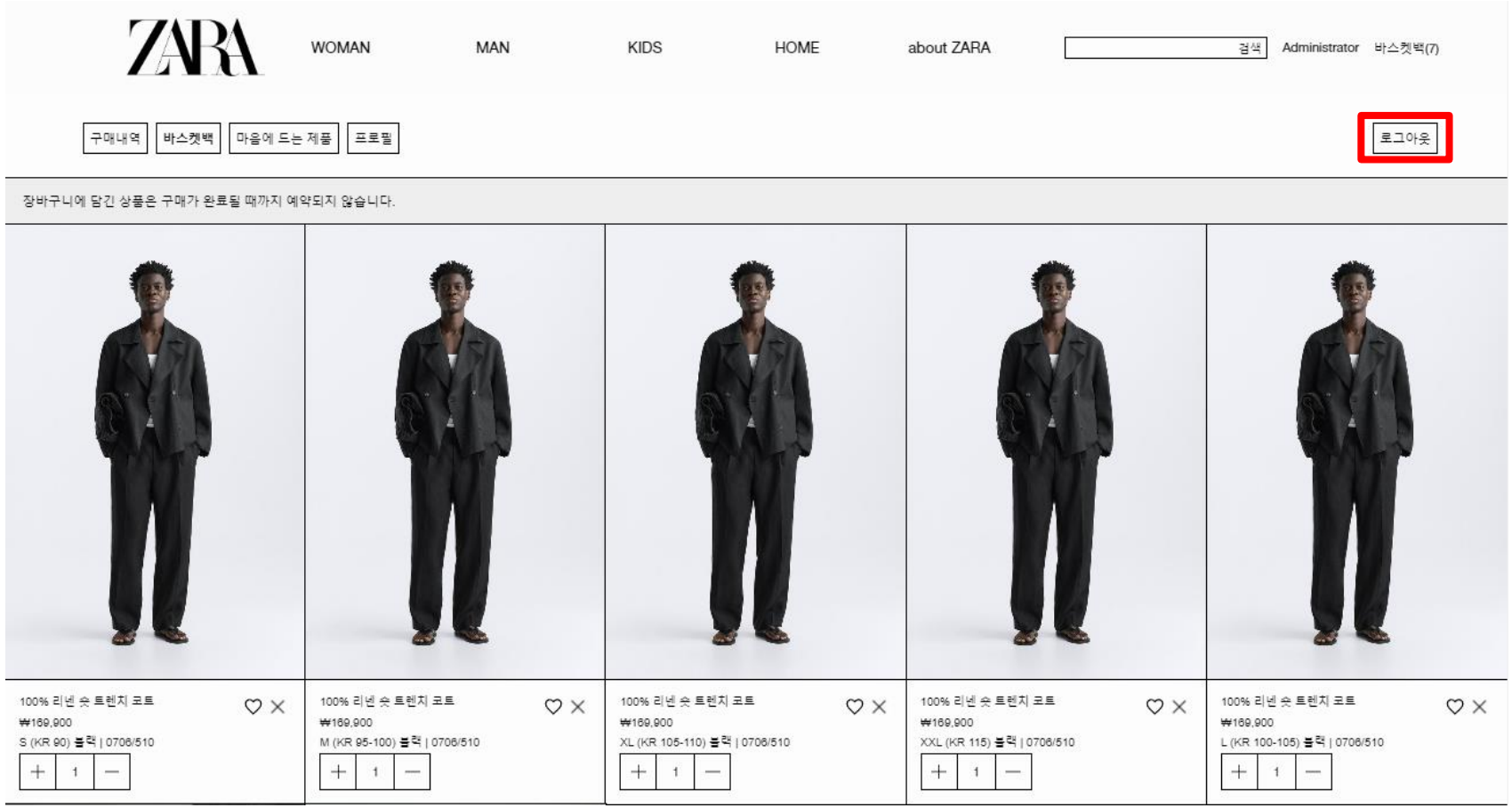
- 원본페이지: 검색창 클릭시->검색페이지로 이동

- 개선페이지: 검색창 직접검색 가능



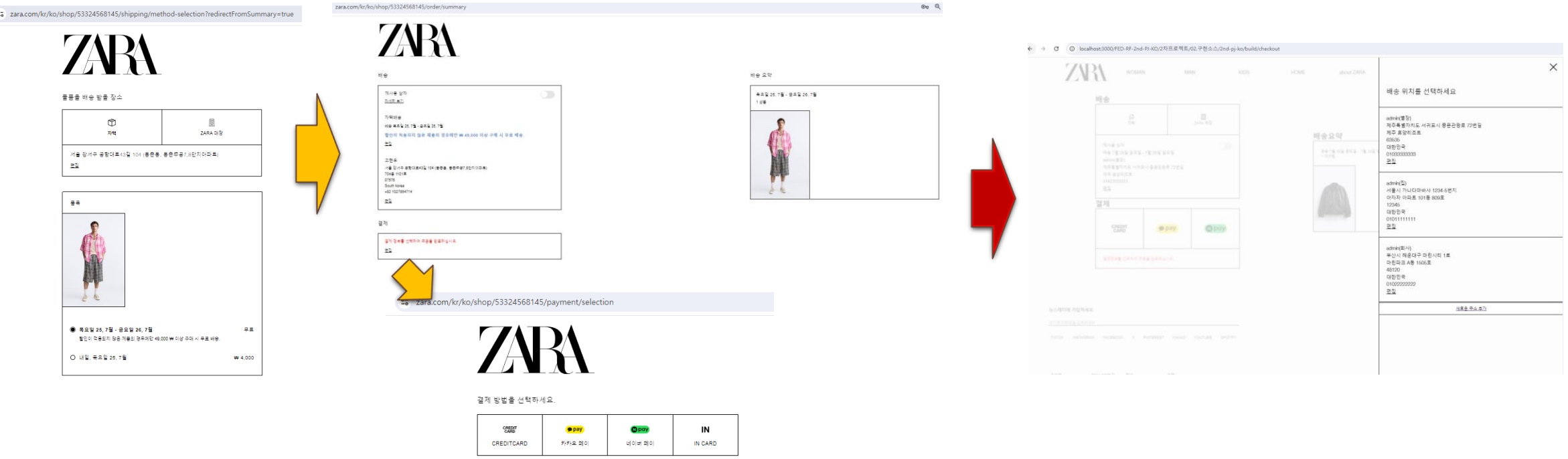
UI&UX개선

- 개선된 사이트: 로그아웃 가능!



UI&UX개선

- 원본사이트: 페이지가 너무 많음
- 개선: 모달창을 열어서 최대한 한페이지에서 처리 가능하게 끔



기술: 커스텀 훅(위시리스트)

```
// 위시리스트 커스텀훅(훅들을 포함해서 재사용할 수 있음, 항상use로 시작해야함)
const useFavoriteFn = () => {

    // 즐겨찾기 목록을 관리하기 위한 상태
    const [favorites, setFavorites] = useState([]);

    // 컴포넌트가 마운트될 때 로컬 스토리지에서 즐겨찾기 데이터를 불러옴
    useEffect(() => {
        // 로컬 스토리지에서 "favorite-data" 키로 저장된 데이터를 가져옴
        const storedFavorites = localStorage.getItem("favorite-data");
        // 저장된 데이터가 있으면 파싱하여 상태에 설정
        if (storedFavorites) {
            setFavorites(JSON.parse(storedFavorites));
        }
    }, []); // 빈 배열을 전달하여 컴포넌트 마운트 시에만 실행

    // 즐겨찾기 토글 함수
    const toggleFavorite = (aaa) => {
        // 현재 즐겨찾기 목록을 복사
        const newFavorites = [...favorites];
        // 현재 아이템이 즐겨찾기에 있는지 확인
        const index = newFavorites.findIndex((item) => item.idx === aaa.idx);

        if (index !== -1) {
            // 이미 즐겨찾기에 있으면 제거
            newFavorites.splice(index, 1);
        } else {
            // 즐겨찾기에 없으면 추가
            newFavorites.push({
                idx: aaa.idx,
                name: aaa.name,
                price: [aaa.price[0], aaa.price[1], aaa.price[2],],
                color: aaa.color,
                isrc: aaa.isrc,
                cnt: 1, // 기본 수량을 1로 설정
            });
        }

        // 새로운 즐겨찾기 목록으로 상태 업데이트
        setFavorites(newFavorites);
        // 업데이트된 즐겨찾기 목록을 로컬 스토리지에 저장
        localStorage.setItem("favorite-data", JSON.stringify(newFavorites));
    };


    // favorites와 toggleFavorite을 반환
    return { favorites, toggleFavorite };
};
```

```
//favorite, 하트버튼 사용을 위한 내가만든 커스텀 훅!
const { favorites, toggleFavorite } = useFavoriteFn();
```

```
{/* 하트버튼:favorite버튼 ********** */}
<div
    className="heartbutton"
    onClick={(e) => {
        if (isMobile) e.stopPropagation();
        toggleFavorite(data);
    }}
>
    {favorites.some((fav) => fav.idx === data.idx) ? (
        <IoMdHeart size={20} />
    ) : (
        <IoMdHeartEmpty size={20} />
    )}
</div>
```

- 검색기능
- 디테일 페이지
- 장바구니, 장바구니 추가 버튼
- 마음에드는 제품, 즐겨찾기 버튼
- CNT 변경에 따른, 총 합계값 변경
- CheckOut페이지에서 주소값 변경, 추가(기존 정보 없을시 추가가능)

  :로컬저장,변경

...화면을 보면서 설명

향후계획

- 메모제이션(상단,메인 영역)***
- 프로필→주소, 이메일, 전화번호, 비밀번호 수정부분 마무리***
- 제품리스트 →버튼기능 생성***
- 게시판구현 → admin으로 로그인시에만 나오는 게시판, 데이터 입력해서 상품에 출력할 수 있게*
- DetailPage →이미지 클릭시 이미지 확대기능**
- 장바구니, 위시리스트 로컬스토리지에 계정분리 해서 저장하기

- PPT*