

React Portfolio Guide Sheet

정보 전달을 주제로 기업형 사이트를 구현했습니다.

보여주어야 할 정보가 명확한 사이트를 만들고자 직관적이고 명확한 레이아웃,
단순한 동선과 주요 정보를 강조할 수 있는 잡지를 디자인 컨셉으로 선정했으며
사이트에 들어가는 description은 패션 잡지 vogue의 웹사이트에서 영감을 받
아 제작되었습니다.

THE FEATURE OF SITE

- Redux로 양방향data관리, Redux-Saga로 api요청
- 게시판data를 localstorage에 저장하고 CRUD기능
- Youtube Api data관리
- Flickr Api data를 가져오고 data 검색 기능
- 페이지 로드 시간을 단축을 위한 이미지캐싱
- 반응형 style적용
- 컴포넌트 재사용 편의성을 위해 풀더 분리



Directory

npx create-react-app <폴더명> 명령어를 통해 설치하여 생성된 public 폴더와 src 폴더에 대한 설명입니다.

정적 자료를 담는 public 폴더와 개발 자료 전반을 담는 src 폴더를 어떻게 나누어 사용하였는지 나타냅니다



Common components

components

- common**
 - 재사용이 가능한 컴포넌트는 common 폴더에서 관리

Header.js	공통 Header component
Layout.js	공통 sub page layout component
Footer.js	공통 Footer component
Menu.js	공통 mobile menu component
Modal.js	공통 pop up modal component

SCSS

- common**
 - common 컴포넌트에 대응하는 stylesheet는 common 폴더에서 관리

_header.scss	Header component style
_sub-common.scss	Layout component style
_footer.scss	Footer component style
_menu.scss	mobile menu component style
_modal.scss	mobile menu component style

Header 공통 Header components

Main page와 Sub page에서 Header의 스타일이 달라지기 때문에 switch를 사용하여 exact 표시된 main type을 먼저 채택하고 그렇지 않으면 sub type을 선택하도록 구현

```
<Switch>
  <Route exact path='/' render={() => <Main menu={menu} />} />
  <Route path='/' render={() => <Header type='[sub]' menu={menu} />} /></Route>
</Switch>
```

tablet 디바이스부터는 bar menu로 submenu로 나타내며 bar icon을 클릭 시 모바일 서브메뉴 패널 토글

VOGUE



```
<Link
  to='/'
  className='hamMenu'
  onClick={(e) => {
    e.preventDefault();
    props.menu.current.setToggle();
  }}
>
  <span>메뉴호출</span>
</Link>
```

VOGUE

About Gallery Board Join Contact



Contents

공통 sub page layout components

VOGUE

Contact me

Social

If you are interested in my portfolio, please email me! instagram
blog

View

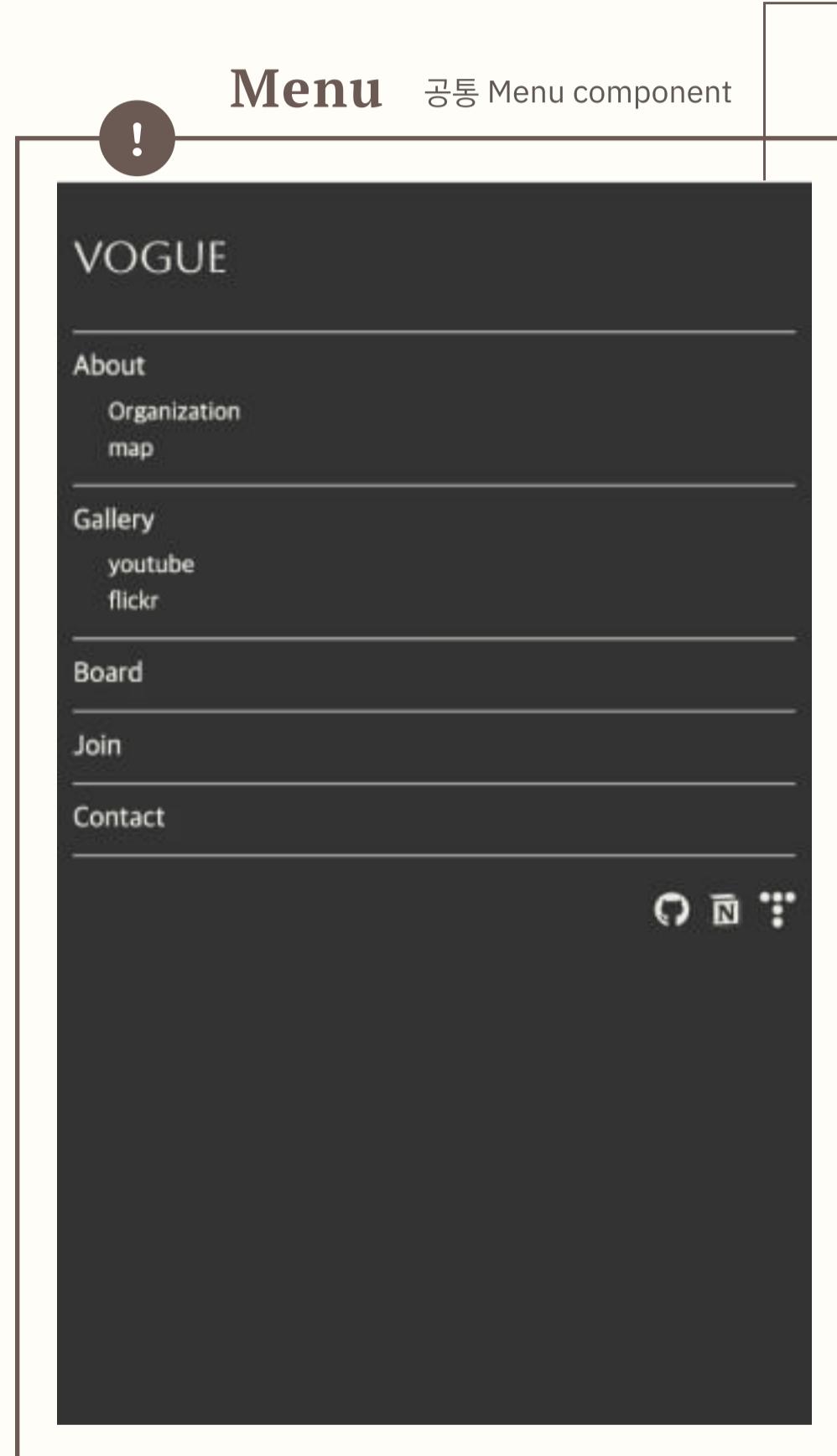
github tistory notion

Copyright © 2023 Yoom ALL Right reserved



Footer 공통 Footer components

Common components



Menu 공통 Menu component

prop을 넘겨서 내부에 있는 HTML 엘리먼트에 접근할 수 있도록 forwardRef를 사용하여 부모 컴포넌트 뿐만 아니라 다른 컴포넌트에서도 접근 가능

```
const Menu = forwardRef((props, ref) => { })
```

리사이즈시 (디바이스 크기가 태블릿보다 커질 때) 자동으로 패널이 닫히도록 구현

```
useEffect(() => {
  window.addEventListener('resize', () => {
    if (window.innerWidth >= 1180) setOpen(false);
  });
}, []);
```

APPLY

App.js에서 호출한 menu컴포넌트에서 전달된 참조객체를 다시 메인 전용 Header컴포넌트에 전달하기 위해 우선 Main 컴포넌트 props로 전달

```
<Switch>
  <Route exact path="/" render={() => <Main menu={menu} />} />
  <Route path="/" render={() => <Header type='sub' menu={menu} />} /></Route>
</Switch>
```

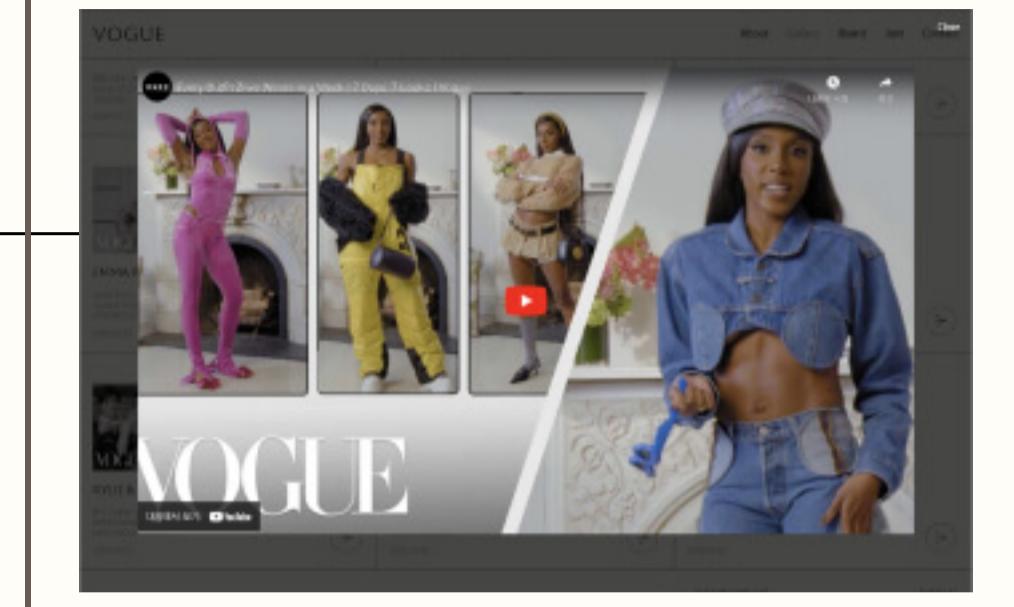
Header 컴포넌트에서 props로 전달받은 참조객체의 setToggle함수 호출

```
<Link
  to '/'
  className='hamMenu'
  onClick={(e) => {
    e.preventDefault();
    props.menu.current.setToggle();
  }}
>
  <span>메뉴호출</span>
</Link>
```

Main.js에 선언된 Header의 특정 값을 직계자식이 아닌 자손 컴포넌트에 전달하기 위해 불필요하게 중간 컴포넌트에 props를 전달 - Prop drilling

```
<Header type='main' menu={props.menu} />
```

Modal

 공통 pop up modal component

prop을 넘겨서 내부에 있는 HTML 엘리먼트에 접근할 수 있도록 forwardRef를 사용하여 부모 컴포넌트 뿐만 아니라 다른 컴포넌트에서도 접근 가능

```
const Modal = forwardRef((props, ref) => { })
```

모달을 여는 state변경함수를 부모 컴포넌트로 전달

```
useImperativeHandle(ref, () => {
  return { open: () => setOpen(true) };
});
```

APPLY

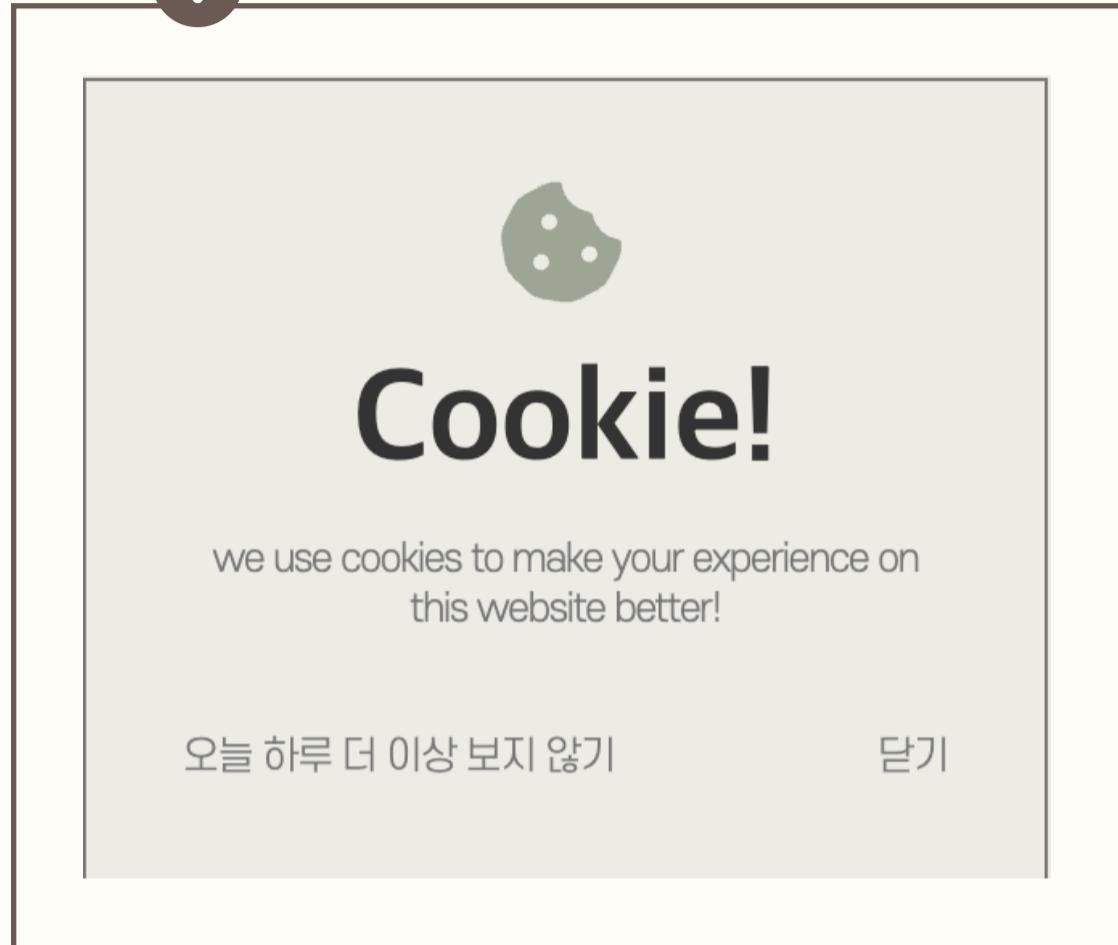
Youtube, Flickr 등등 모달이 필요한 곳에서 ref값을 받음

```
<Modal ref={open}>
  <iframe title={Vids[Index].id} src={`https://www.youtube.com/embed/${Vids[Index].snippet.resourceId.videoId}`}></iframe>
</Modal>
```

Common components

Cookie popup

cookie popup component



components > common > CookiePopUp.js

styled-components 적용하여 모달 스타일링

```
const ModalContainer = styled.div`  
  position: fixed;  
  top: 100px;  
  right: 10px;  
  z-index: 1000;  
  width: 300px;  
  ...  
`;
```

props로 모달 닫힘 버튼, 만료일 넘겨줌

```
function Cookie({ closeModal, closeModalUntilExpires }) {  
  return (  
    ...생략  
    <ModalCloseWrapper>  
      <p onClick={closeModalUntilExpires}>오늘 하루 더 이상 보지 않기</p>  
      <p onClick={closeModal}>닫기</p>  
    </ModalCloseWrapper>  
    ...생략  
  )  
}
```

App.js에서 적용

react-cookie 라이브러리를 이용하여 쿠키에 접근하여 cookie 모달, 쿠키를 가지고 있는지의 유무를 useState를 통해 상태 감지

```
const [openCookie, setOpenCookie] = useState(true);  
const [hasCookie, setHasCookie] = useState(true);  
const [appCookies, setAppCookies] = useCookies();
```

getExpiredDate 함수에서 현재 시간에서 해당 일수를 더한 시기를 반환하여 만료시기를 정한 후 closeModalUntilExpires 함수에서 만료일 1일인 쿠키 생성, 모달 닫는 버튼 구현

```
const getExpiredDate = (day) => {  
  const date = new Date();  
  date.setDate(date.getDate() + day);  
  return date;  
};  
  
const closeModalUntilExpires = () => {  
  if (!appCookies) return;  
  const expires = getExpiredDate(1);  
  setAppCookies('MODAL_EXPIRES', true, { path: '/', expires });  
  setOpenCookie(false);  
};
```

해당 쿠키가 존재하면 모달 false, 없으면 모달 true 되도록 구현

```
useEffect(() => {  
  if (appCookies['MODAL_EXPIRES']) return;  
  console.log(appCookies['MODAL_EXPIRES']);  
  setHasCookie(false);  
}, [appCookies]);
```

모달의 유무와 쿠키의 유무를 감지 하며 props로 받은 setOpenCookie(false) 함수, closeModalUntilExpires 함수를 받아 처리

```
{openCookie && !hasCookie && (  
  <Cookie closeModal={() => setOpenCookie(false)}>  
    closeModalUntilExpires={closeModalUntilExpires} />  
)}
```

버튼 클릭 시 scroll 값이 맨 위로 가도록 구현

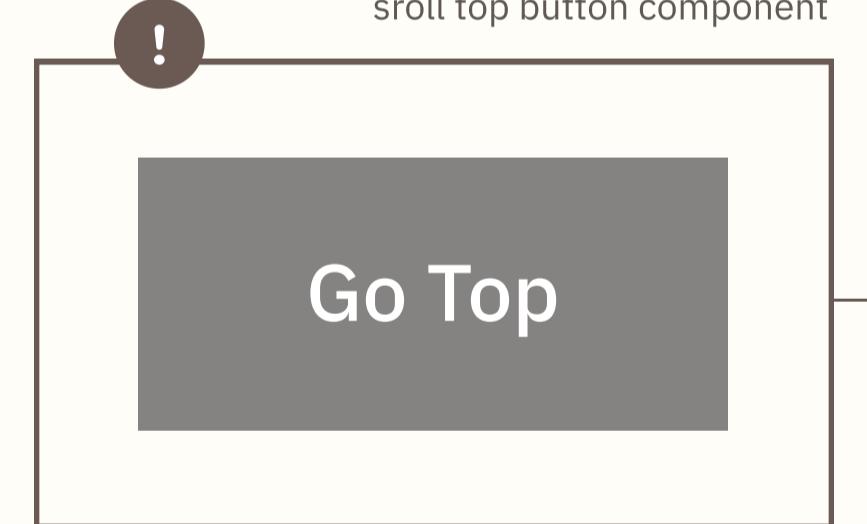
```
const scrollToTop = () => {  
  window.scroll({  
    top: 0,  
    behavior: 'smooth',  
  });  
};
```

scroll값이 300이 넘어갈 때만 버튼이 보이도록 구현

```
const handleShowButton = () => {  
  if (window.scrollY > 300) {  
    setShowButton(true);  
  } else {  
    setShowButton(false);  
  }  
};
```

Top Button

scroll top button component



Common Scripts

components

assets

프로젝트에서 사용되는 라이브러리와 커스텀혹은 assets에서 관리

- ☰ Anime.js animation 구현 라이브러리
- ☰ Scroll.js useScroll 커스텀훅

Anime.js

불필요한 리소스 낭비 방지 애니메이션 라이브러리

```
export default class Anim {
  run(time) {
    let timelast = time - this.startTime;
    let progress = timelast / this.option.duration;
    progress < 0 && (progress = 0);
    progress > 1 && (progress = 1);
    progress < 1
      ? requestAnimationFrame((time) => this.run(time))
      : this.option.callback && setTimeout(this.option.callback, 0);
    let result = this.currentValue + (this.option.value -
    this.currentValue) * progress;

    ...
  }
}
```

requestAnimationFrame을 사용하여 불필요한 리소스 낭비를 줄여줌
자기기 자신을 호출하는 재귀함수 형태로 반복기능을 구현해주고 무한루프에 빠지지 않도록 조건문으로 반복기능이 끝나는 조건도 같이 설정

requestAnimationFrame에 의해서 각 반복 사이클마다 호출되는 누적시간을 구하고 그 누적시간을 활용하여 특정 시간대비 진행률 계산해주고 progress를 이용하여 원하는 시간, 원하는 수치를 모션처리해줌

Anime.js 적용방법

```
new Anim(window, {
  prop: 'scroll',
  value: pos.current[idx],
  duration: speed.current,
});
```

UseScroll customhook

재사용성을 위한 props 값 설정

```
const useScrollFadeIn = (direction = 'up', duration = 2, delay = 0) => {};
```

direction : 엘리먼트가 나오는 방향

duration: 애니메이션 동작시간

delay : 애니메이션 지연시간

Element 감지하기

```
useEffect(() => {
  let observer;
  const { current } = element;

  if (current) {
    observer = new IntersectionObserver(handleScroll, { threshold: 0.7 });
    observer.observe(current);
  }

  return () => observer && observer.disconnect();
}, [handleScroll]);
```

리플로우가 발생하는 getBoundingClientRect()을 사용하여 element의 값을 구하지 않고 Intersection Observer를 사용하여 observer에 감지가 되고, 70% (option : threshold) 가 보였을때 'handleScroll' 을 실행하도록 설정

Hook 적용방법

```
<div {...useScrollFadeIn("up", 1, 0)}></div>
```

Common Style

_color.scss

색상의 용도에 따라 변수를 설정하여 프로젝트에 대입

```
// main color
$backgroundColor : #ECECE3;

// second color
$btnColor : #D9D9CE;

//point color
$hoverColor: #9DA592;
$errorColor : red;

//grayScale
$white: #ffffff;
$black: #000000;
$textColor: #333333;
$borderColor: #777777;
```

_layout.scss

반응형 대응 브라우저 사이즈

```
$sLaptop: 1024px;
$Ltablet: 992px;
$tablet: 778px;
$mobile: 576px;
$sMobile: 425px;
```

_font.scss

프로젝트에 사용될 font import 및 변수 설정

```
// font
$titleFont : 'Aboreto';
$enFont : 'Libre+Bodoni';
$koFont : 'S-CoreDream-3Light';
```



프로젝트에서 공용으로 사용되는
stylesheet는 common에서 관리

- _color.scss 프로젝트에서 중복사용되는 컬러를 변수화
- _font.scss 프로젝트에서 사용되는 폰트 선언 및 변수화
- _layout.scss 디바이스 별 대응을 위한 반응형 px 변수화
- _common.scss 공통 UI(버튼) style sheet
- _reset.scss 기존 style 초기화를 위한 sheet
- _mixin.scss 반복적으로 재사용할 style 선언

_reset.scss

공통으로 사용되는 초기화 style 적용

```
.hidden {
    position: absolute;
    top: -999999999999px;
    opacity: 0;
}
```

_mixin.scss

section의 title, text 등 반복적으로 재사용할 style 선언

```
@mixin sectionTitle (
    $fz: 60px,
    $letterS: -.56px,
    $lh: 1,
    $txtTrans : uppercase,
    $fontColor: #333,
    $fontWeight: 400,
) {
    font-family: $titleFont;
    font-size: $fz;
    letter-spacing: $letterS;
    line-height: $lh;
    text-transform: $txtTrans;
    color: $fontColor;
    font-weight: 400;
}
```

_common.scss

공통으로 사용되는 UI 요소의 style 적용

ex) button 1

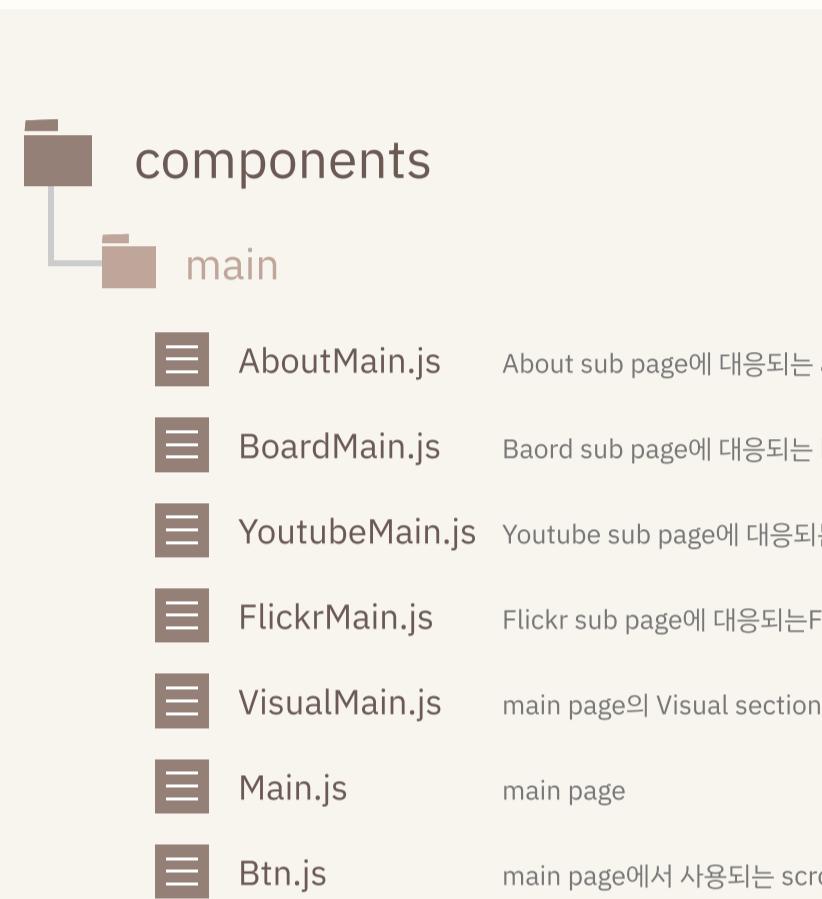
view MORE

ex) button 2

view MORE →

Information from the React Portfolio

Main Page



main visual 영역

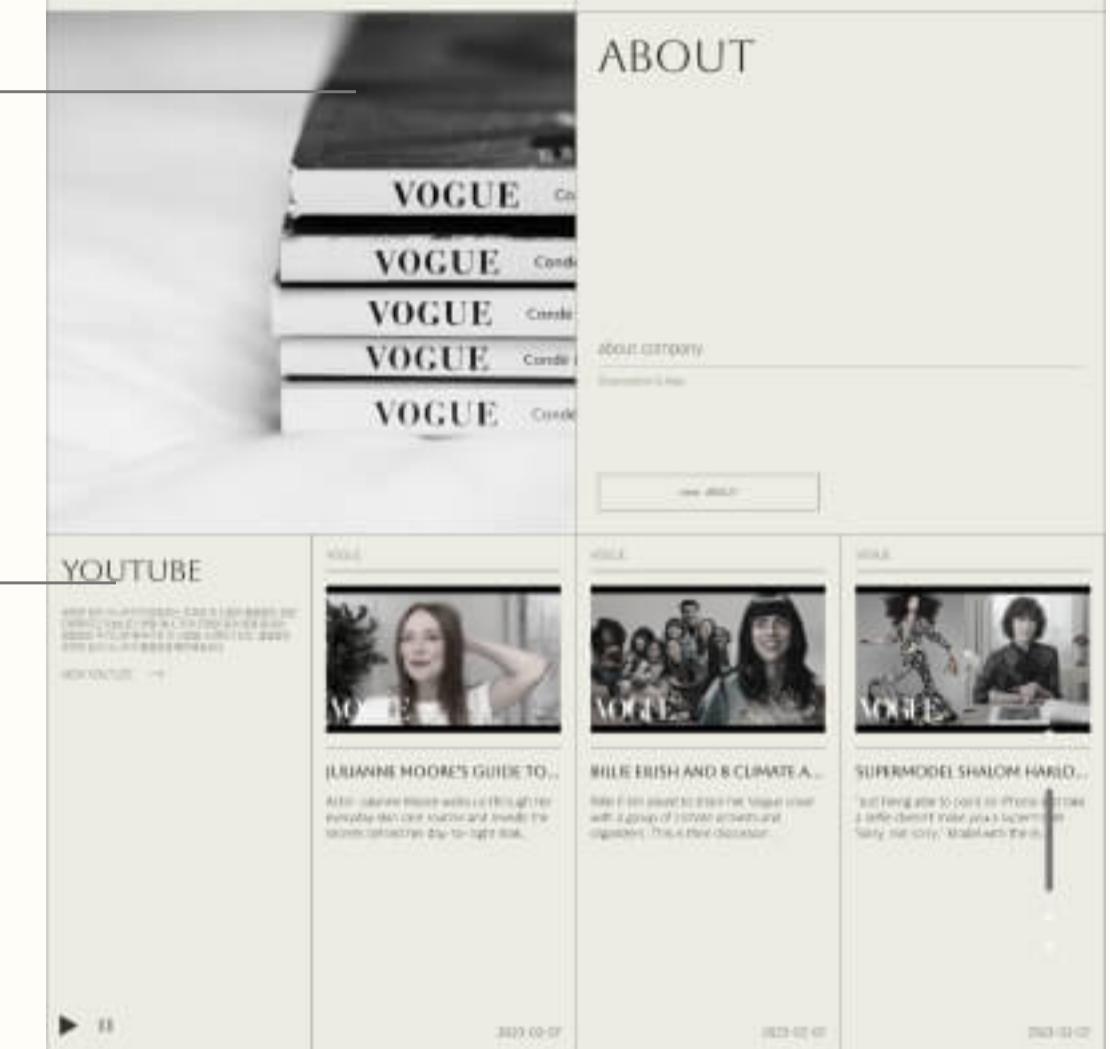
portfolio의 main visual 영역
useScroll hook을 이용한 텍스트 인터렉션 적용



ABOUT

about contents 영역

about page contents 영역
useScroll hook을 이용한 텍스트 인터렉션 적용
button 클릭시 about page로 이동



youtube contents 영역

youtube contents 영역

1. 유튜브 페이지에 있는 데이터를 메인 화면에 보여주기 위해서 Redux-saga를 활용. youtube api를 store에 저장 후 메인 화면과 서브 페이지에 동일한 데이터를 사용하여 화면에 보여줌.
2. thumbnail 클릭 시 popup modal 나오도록 구현
3. swiper plugin을 적용하여 슬라이더를 구현했으며 autoplay, loop를 이용하여 연속 기능을 부여하. 또한 popup modal이 나올 시 재생이 일시정지 되는 로직을 구현함



flickr contents 영역

flickr contents 영역

1. flickr 서브페이지에 있는 데이터를 메인 화면에 보여주기 위해서 Redux-saga를 활용. flickr api를 store에 저장 후 메인 화면과 서브 페이지에 동일한 데이터를 사용하여 화면에 보여줌.
 2. thumbnail 클릭 시 popup modal 나오도록 구현

메인 페이지 스크롤시 불필요하게 재랜더링되는 main section
컴포넌트들을 memo로 메모이제이션 해줌

예시 ->

```
export default memo(BoardMain);
```

Main Page - Btn.js



Btn.js

section별 offsetTop값을 구하여 스크롤 효과 적용 버튼 활성화 함수에서 클릭시 섹션의 offsetTop값 + 섹션 영역이 3분의 2이상일 때 버튼이 활성화 되도록 설정, 부모컴포넌트로부터 전달받은 setScrolled함수로 현재 내부적으로 만들어지고 있는 scroll거리값을 부모 Scrolled state에 옮겨담음, Anime.js plugin을 활용하여 스크롤 효과 적용. 버튼 클릭시 해당 index번째의 section으로 이동할 수 있도록 index값을 구해주고 duration을 설정하여 애니메이션의 지속시간또한 설정

useCallback을 이용해서 해당 getPos, activation함수를 미리 메모이제이션하여 컴포넌트가 재랜더링 되더라도 미리 메모이제이션 처리한 함수를 재사용하도록 설정

Main.js에 적용

Btns에 state변경함수를 전달해서 Bns 내부적으로 만들어지는 값을 부모 State에 저장하고 저장된 State값을 다시 자식인 다른 컴포넌트에 전달하는 식으로 Btn 컴포넌트 scroll값을 다른 컴포넌트에 전달함

```
const getPos = useCallback(() => {
  pos.current = [];
  const secs =
    btnRef.current.parentElement.querySelectorAll('.myScroll');
```

```
for (const sec of secs) {
  if (matchMedia('screen and (min-width: 1024px)').matches) {
    pos.current.push(sec.offsetTop - 79);
  } else {
    pos.current.push(sec.offsetTop - 59);
  }
}
```

```
setPos(pos.current);
}, [setPos]);
```

```
new Anim(window, {
  prop: 'scroll',
  value: pos.current[idx],
  duration: speed.current,
});
```

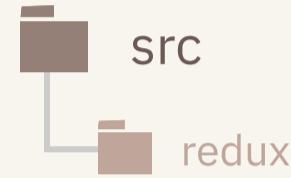
```
function Main(props) {
  const [Scrolled, setScrolled] = useState(0);
  const [Pos, setPos] = useState([]);
  return (
    <main id='main'>
      <div className='wrap'>
        <VisualMain />
        <AboutMain Scrolled={Scrolled} Pos={Pos[1]} />
        <YoutubeMain />
        <FlickrMain />
        <BoardMain />
        <Bsns setScrolled={setScrolled} setPos={setPos} />
      </div>
    </main>
  );
}
```

```
const activation = useCallback(() => {
  const btns = btnRef.current.children;
  const secs =
    btnRef.current.parentElement.querySelectorAll('.myScroll');
  const scroll = window.scrollY;
  const base = -window.innerHeight / 3;

  setScrolled(scroll);

  pos.current.forEach((pos, idx) => {
    if (scroll >= pos + base) {
      for (const btn of btns) btn.classList.remove('on');
      for (const sec of secs) sec.classList.remove('on');
      btns[idx].classList.add('on');
      secs[idx].classList.add('on');
    }
  });
}, [setScrolled]);
```

Redux - saga



actionType.js	각 데이터 별로 활용한 문자열 액션 타입을 변수처럼 모아놓은 객체
reducer.js	전역 state의 데이터를 변경하는 함수
saga.js	Youtube sub page에 reducer에 추가 가능 확장을 위한 미들웨어 youtube section
api.js	외부 비동기 데이터 호출문을 이 곳에 컴포넌트 외부 파일로 따로 관리
store.js	state가 저장될 전역 공간

ActionType.js

Youtube page와 Flickr page에서 사용될 데이터의 활용할 문자열 액션 타입을 지정
예시는 youtube data

```

export const YOUTUBE = {
  start: 'YOUTUBE_START',
  success: 'YOUTUBE_SUCCESS',
  fail: 'YOUTUBE_FAIL',
};

```

Reducer.js

컴포넌트에서 넘어오는 action type에 따라 해당 요청이 넘어오면 작업을 saga.js로 넘김
saga가 해당 액션요청을 받아서 비동기 서버통신 함수 호출해서 결과값에 따라 액션객체를 다시 리듀서에 전달

```

const youtubeReducer = (state = { youtube: [] }, action) => {
  switch (action.type) {
    case types.YOUTUBE.start:
      return state;
    case types.YOUTUBE.success:
      return { ...state, youtube: action.payload };
    case types.YOUTUBE.fail:
      return { ...state, youtube: action.payload };
    default:
      return state;
  }
};

```

Api.js

Youtube data와 Flickr data의 비동기 데이터 호출문 option.type을 지정하여 RESTful Api를 다양하게 활용
순수함수 형태로 비동기 데이터를 호출한다
예시는 Flickr data

```

export const fetchFlickr = async (opt) => {
  const base = 'https://www.flickr.com/services/rest/?format=json&nojsoncallback=1';
  const method_interest = 'flickr.interestingness.getList';
  const method_search = 'flickr.photos.search';
  const method_user = 'flickr.people.getPhotos';
  const key = 'api key';
  const per_page = 30;

  let url = "";

  if (opt.type === 'interest')
    url = `${base}&api_key=${key}&method=${method_interest}&per_page=${per_page}`;
  if (opt.type === 'search')
    url = `${base}&api_key=${key}&method=${method_search}&per_page=${per_page}&tags=${opt.tags}`;
  if (opt.type === 'user')
    url = `${base}&method=${method_user}&api_key=${key}&per_page=${per_page}&user_id=${opt.user}`;

  return await axios.get(url);
};

```

Redux - saga



src

redux

- actionType.js
- reducer.js
- saga.js**
- api.js
- store.js

saga.js

1. callYoutube 함수로 컴포넌트로 들어온 YOUTUBE_START라는 액션요청을 리듀서 함수를 통해서 전달 받으면 유튜브 호출함수를 실행,
2. returnYoutube로 유튜브 데이터 호출한 뒤, 결과값을 가지고 다시 새로운 액션객체 반환 후 리듀서에 전달
3. 위의 함수들을 호출해주는 함수를 만든뒤 최종적으로 export

```

function* callYoutube() {
    yield takeLatest(types.YOUTUBE.start, returnYoutube);
}

function* returnYoutube() {
    try {
        const response = yield call(fetchYoutube);
        yield put({ type: types.YOUTUBE.success, payload: response.data.items });
    } catch (err) {
        yield put({ type: types.YOUTUBE.fail, payload: err });
    }
}

export default function* rootSaga() {
    yield all([fork(callYoutube), fork(callFlickr)]);
}
    
```

store.js

sagaMiddleware함수를 활성화해주고
store공간에 reducer데이터를 전달할때 sagaMiddleware 설정, createStore의 2번째 인자에
applyMiddlewares를 넣음으로써 sagaMiddleware가 작동하도록 함
미들웨어로 연결된 saga에 rootSaga함수 적용 후
saga 미들웨어가 최종 적용된 reducer데이터를 store에 저장하고 해당 store를 export함

```

const sagaMiddleware = createSagaMiddleware();
const store = createStore(reducers, applyMiddleware(sagaMiddleware));
sagaMiddleware.run(rootSaga);
export default store;
    
```

Apply - ex)youtube data

index.js에 provider 컴포넌트를 이용하여 store를 연동하면 store데이터가 App컴포넌트를 통해서 전역에 전달

```

<Provider store={store}>
    <App />
</Provider>
    
```

메인페이지의 youtube section에서 전역에 있는 store데이터를 useSelector로 호출

```
const { youtube } = useSelector((store) => store.youtubeReducer);
```

youtube sub page에서 전역에 전달되는 store데이터를 호출

```
const Vids = useSelector((store) => store.youtubeReducer.youtube);
```

전역으로 관리해주기 위해 App.js에서 dispatch로 상태 변경

```

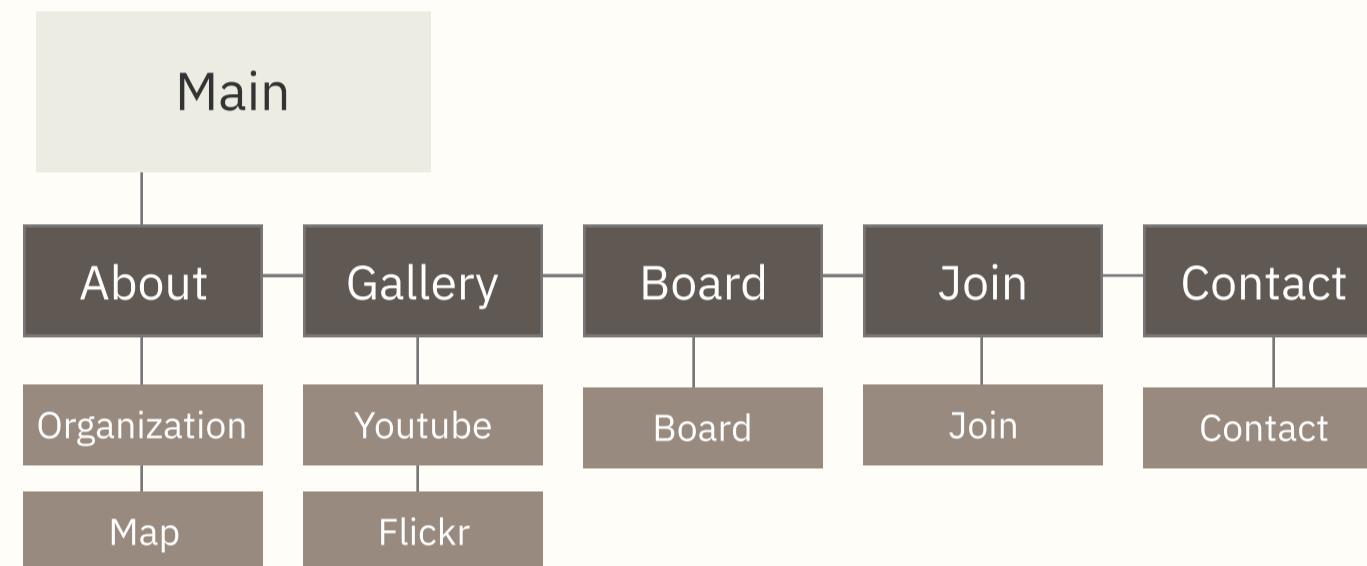
const dispatch = useDispatch();

useEffect(() => {
    dispatch({ type: types.YOUTUBE.start });
}, [dispatch]);
    
```

Information from the React Portfolio

Sub page

Sub page Thesaurus



Sub Page Header 영역

page description 영역

2 depth 메뉴 영역

! ABOUT

Organization / 조직도

Vogue의 새로운 사업 부문인 CCL(Creative Content Lab.) 팀은 두산타이거즈가 보유한 국내 최고의 컨텐츠 혁신 역량을 기반으로, 브랜드의 가치를 높여주는 커뮤니케이션 통합 솔루션을 제시하고 있습니다.

organization map

```
function Layout(props) {
  return (
    <main className={`subPage ${props.name}`}>
      <div className='wrap'>
        <div className='subContents'>
          <header className={`subHeader ${props.name}`}>
            ...생략
          </header>
        </div>
      </div>
    </main>
  )
}
```

Sub Page는 Layout.js 컴포넌트를 만들어 서브 페이지의 공통 레이아웃을 관리
props로 값 전달



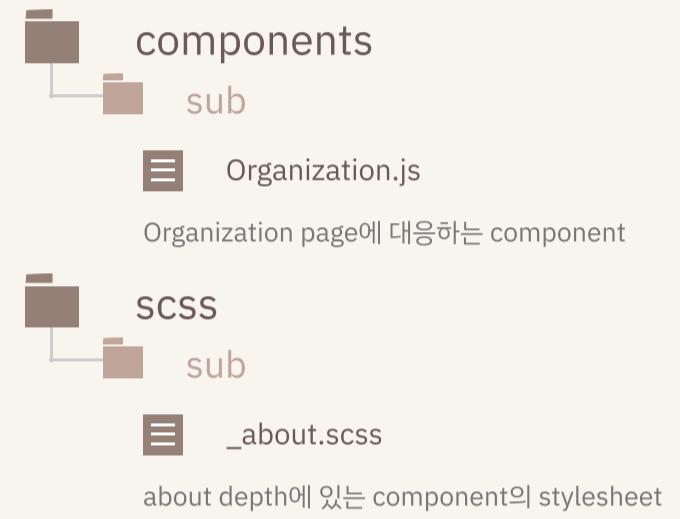
MOTTO

There is no use whatever trying to help people who do not help themselves. You cannot push anyone up a ladder unless he be willing to climb himself.

- Andrew Carnegie

Sub page Layout

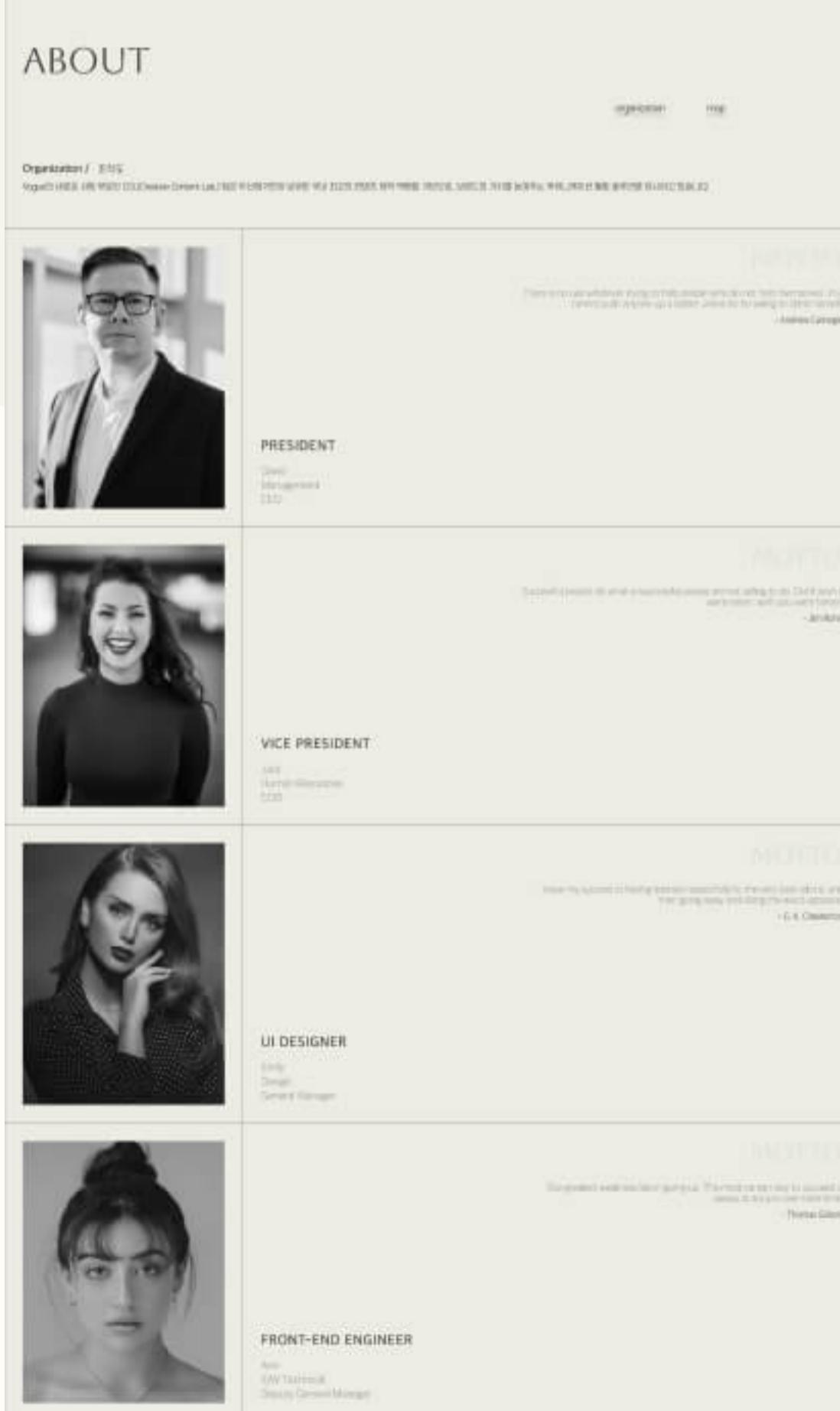
About - Organization.js



Organization.js

기업의 구성원을 나타내는 조직도 페이지

1. Json 파일을 만들어 db 관리
2. Axios로 데이터 호출
3. 환경 변수를 이용하여 public 폴더에 접근



Organization.js

데이터를 json 파일로 직접 만들어 axios로 호출.
json 파일은 public 폴더에 저장 후 환경변수 process.env.PUBLIC_URL
를 사용하여 가져옴

```
useEffect(() => {
  axios.get(`${process.env.PUBLIC_URL}/DB/member.json`).then((json) => {
    setMembers(json.data.members);
  }, []);
});
```

Public > db > member.json

```
"members": [
  {
    "name": "David",
    "department": "Management",
    "position": "President",
    "rank": "CEO",
    "pic": "member01.jpg",
    "wise": "There is no use whatever trying to ....",
    "author": "Andrew Carnegie"
  }
]
```

About - Map.js

components

sub

Map.js

Map page에 대응하는 component

SCSS

sub

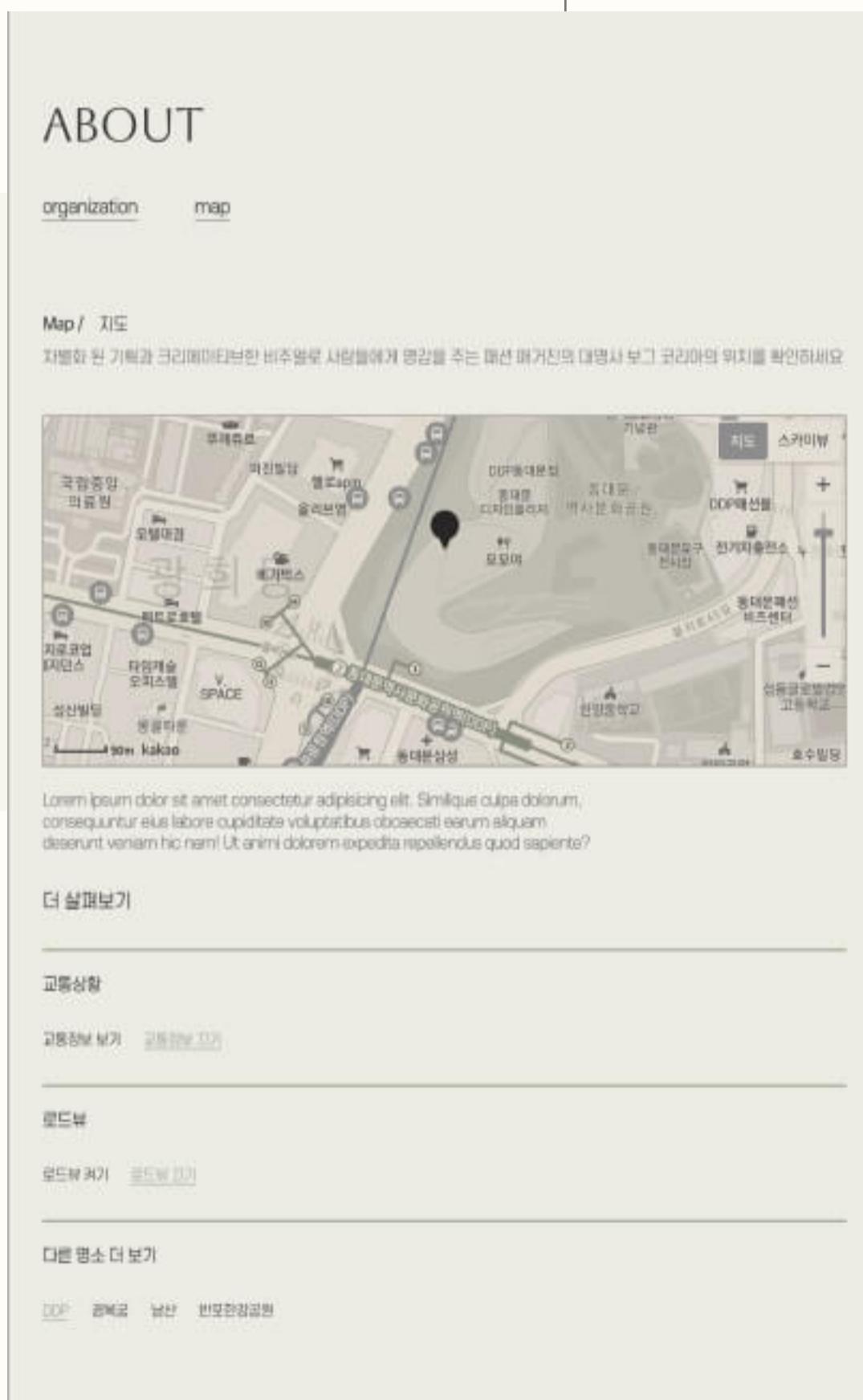
_about.scss

about depth에 있는 component의 stylesheet

Map.js

기업의 위치 및 교통 정보를 담은 페이지

1. Kakao map Api 호출하여 기업의 위치 및 교통정보 표시
2. 교통정보 on/off , 로드뷰 on/off 기능
3. 지점 버튼 클릭시 해당 위치로 이동
4. 지도 타입 컨트롤, 지도 확대축소 기능
5. 페이지 상하 이동 스크롤과 지도 이동 스크롤 중복을 방지하기 위해 지도 스크롤 이동 불가 기능
6. 화면 리사이즈시, 마커 중앙 표시 유지 기능



Map.js

지도가 표시 될 가상 둘을 useRef를 사용하여 참조 객체로 담아줌.

윈도우 객체에서 kakao객체를 비구조화할당으로 바로 할당할 수 있도록 함.

맵인스턴스 컴포넌트내에서 자유롭게 사용하기 위한 state생성

지도에 표시될 기능들의 상태를 useState로 관리

```
const container = useRef(null);
const { kakao } = window;
const [Location, setLocation] = useState(null);
const [Index, setIndex] = useState(0);
const [Traffic, setTraffic] = useState(false);
const [Road, setRoad] = useState(false);
```

첫번째 렌더링 사이클시 Location값이 비어있을때는 실행하지 않고 해당 값이 있을때만 실행되도록 하며 Optional Chaining을 사용하여 객체값이 있을때에만 해당 객체에 종속된 메서드가 호출되도록 함 (로드뷰도 동일)

```
useEffect(() => {
  Traffic
    ? Location?.addOverlayMapTypeId(kakao.maps.MapTypeId.TRAFFIC)
    : Location?.removeOverlayMapTypeId(kakao.maps.MapTypeId.TRAFFIC);
}, [Traffic, kakao, Location]);
```

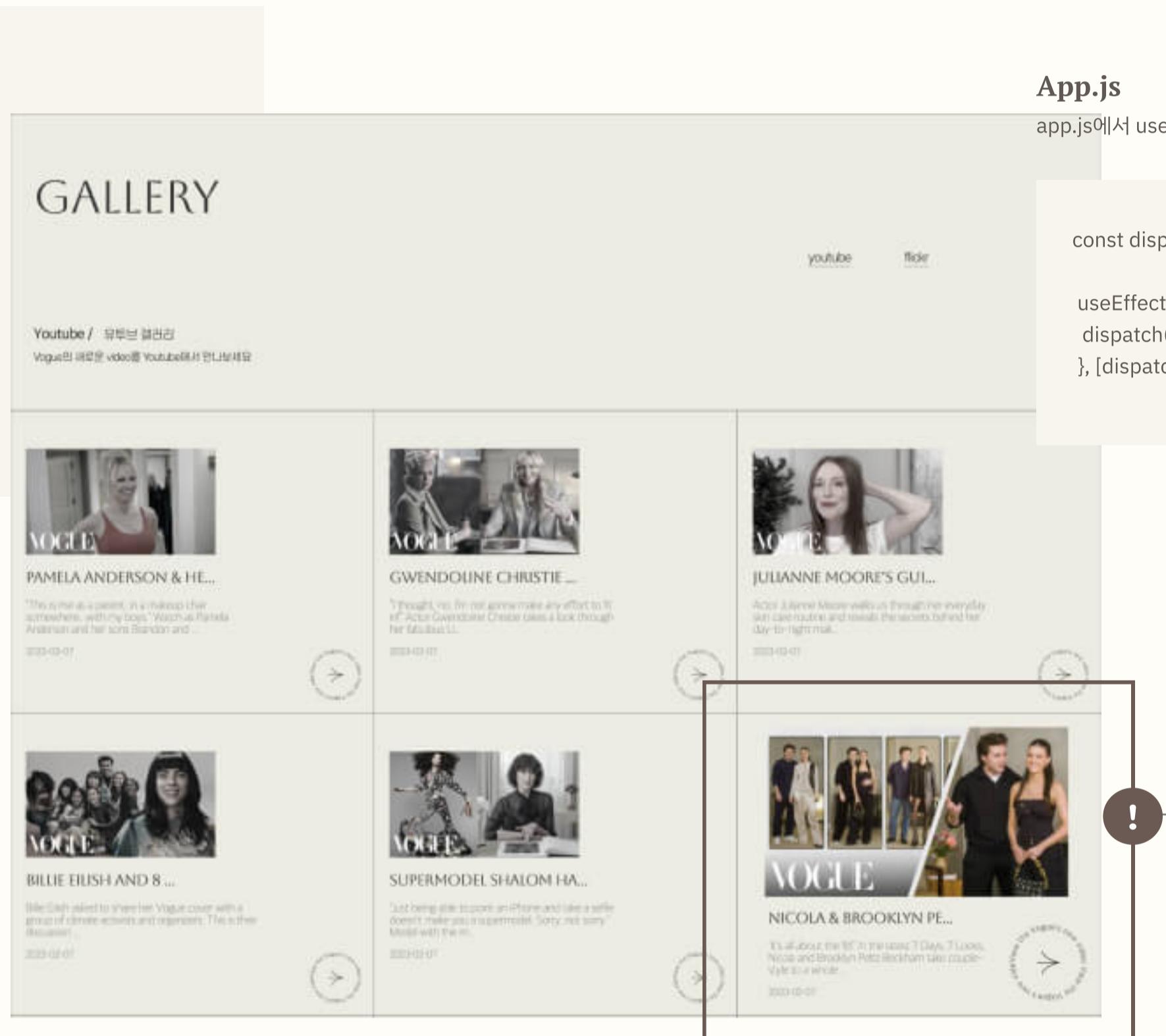
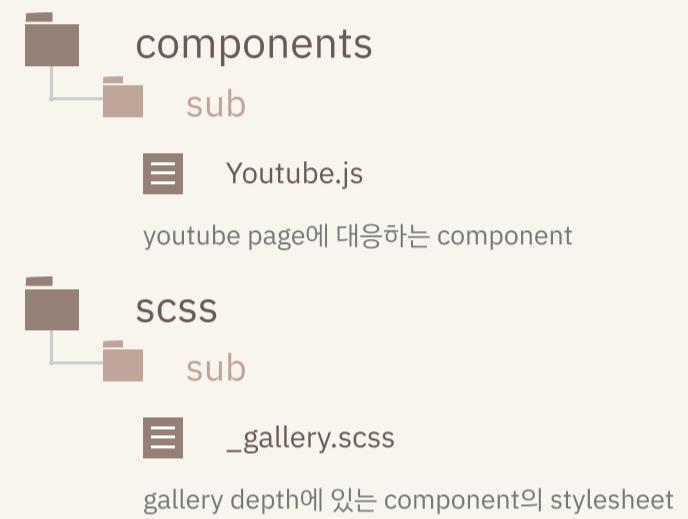
브라우저 리사이즈 될 때마다 마커가 가운데로 위치하도록 설정

해당 컴포넌트 언마운트시 setCenter함수 이벤트를 제거하여 불필요한 렌더링을 방지
페이지 상하 이동 스크롤과 지도 이동 스크롤 중복을 방지

```
useEffect(() => {
  container.current.innerHTML = "";
  ... 생략
  const setCenter = () => {
    mapInstance.setCenter(info[Index].latlng);
  };
  mapInstance.setZoomable(false);
  window.addEventListener('resize', setCenter);
  return () => {
    window.removeEventListener('resize', setCenter);
  };
}, [Index]);
```

Information from the React Portfolio

Gallery- Youtube.js



Youtube.js

데이터는 redux-saga를 이용하여 전역으로 관리해주기 때문에 youtube page에서는 useSelector로 데이터의 상태값을 불러옴

```
const Vids = useSelector((store) => store.youtubeReducer.youtube);
```

App.js

app.js에서 useDispatch로 상태값을 변경해준다.

```
const dispatch = useDispatch();

useEffect(() => {
  dispatch({ type: types.YOUTUBE.start });
}, [dispatch]);
```

Hover Effect

리스트 hover 시 thumbnail의 filter를 조정하여 선명도 조정, transform의 scale 효과를 이용하여 커지도록 구현함
modal 버튼에 rotate animation을 주어 text가 회전하도록 구현함

Gallery- Flickr.js

components

sub

Flickr.js

youtube page에 대응하는 component

SCSS

sub

_gallery.scss

about depth에 있는 component의 stylesheet

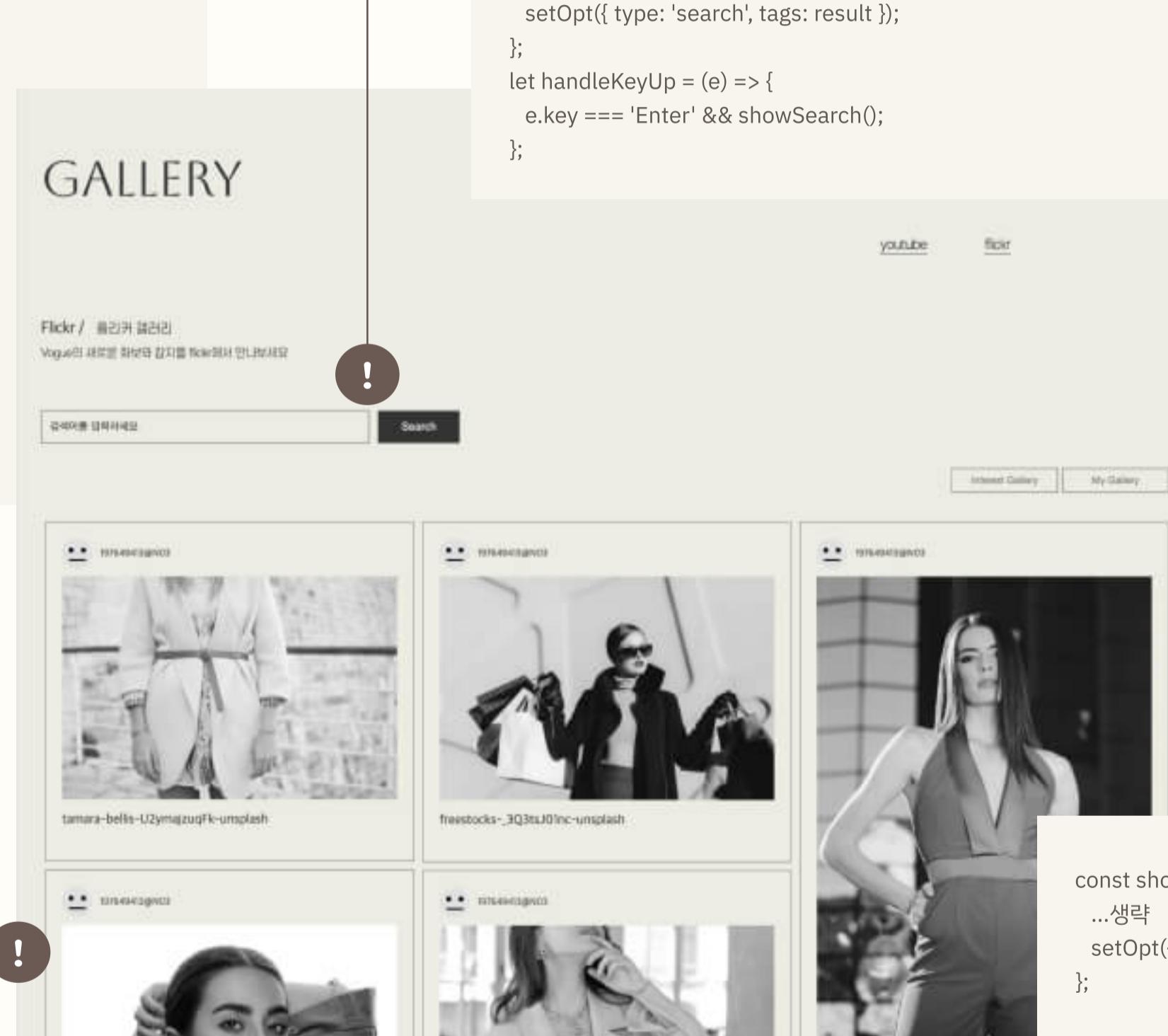
Flickr.js

기업 사진 gallery 페이지

1. Flickr data를 이용한 페이지 구성
2. redux saga를 통한 데이터 상태관리
3. 데이터 검색 기능 추가
4. masonry plugin 적용
5. 데이터 호출되는 시간동안 로딩 기능 추가
6. 이미지 클릭 시 modal 호출

Masonry Layout

Masonry Layout 지정 후 스타일 적용



Search input

Flickr api의 search method를 이용하여 search 기능 구현
trim() 메서드를 이용하여 검색어에 공백 제거.
handleKeyUp 함수를 만들어 키보드 enter시에도 검색 가능하도록 구현

```
const showSearch = () => {
  const result = input.current.value.trim();
  ...생략
  setOpt({ type: 'search', tags: result });
};

let handleKeyUp = (e) => {
  e.key === 'Enter' && showSearch();
};
```

Flickr의 상태값으로 변경하기 위해 useDispatch사용

```
useEffect(() => {
  dispatch({ type: types.FLICKR.start, Opt });
}, [Opt, dispatch]);
```

Sort Btn

Flickr api의 인기게시판, 내 게시판 메서드를 이용하여 구현
setOpt으로 상태 값 지정하고

```
const showInterest = () => {
  ...생략
  setOpt({ type: 'interest' });
};
```

```
const showMine = () => {
  ...생략
  setOpt({ type: 'user', user: 'userId' });
};
```

Information from the React Portfolio

Board

components

sub

Board.js

board page에 대응하는 component

SCSS

sub

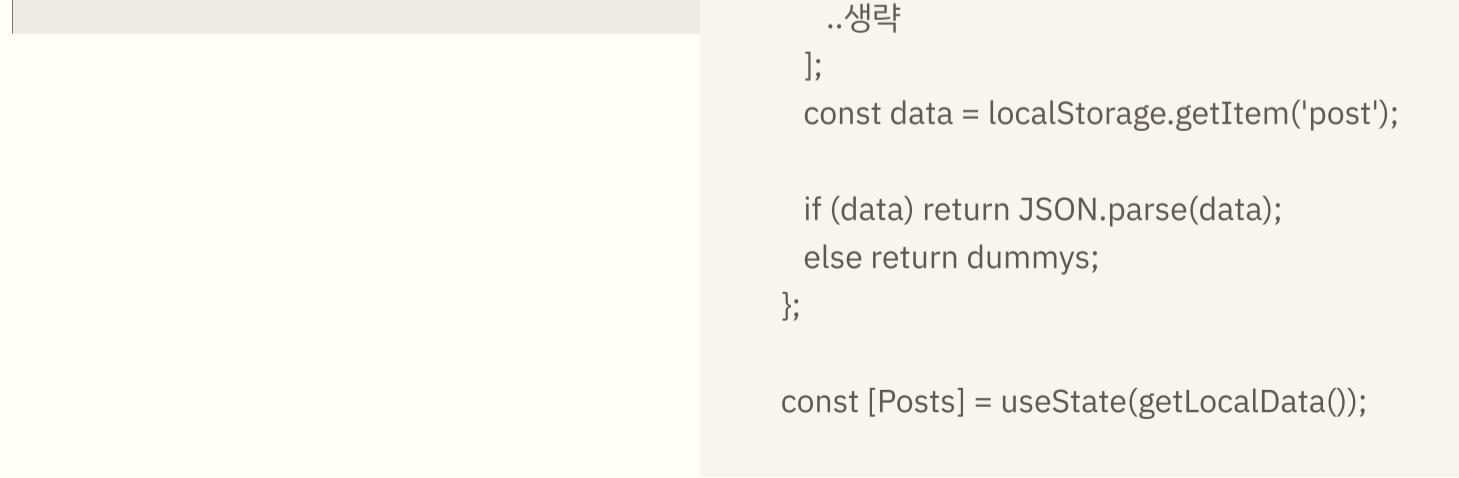
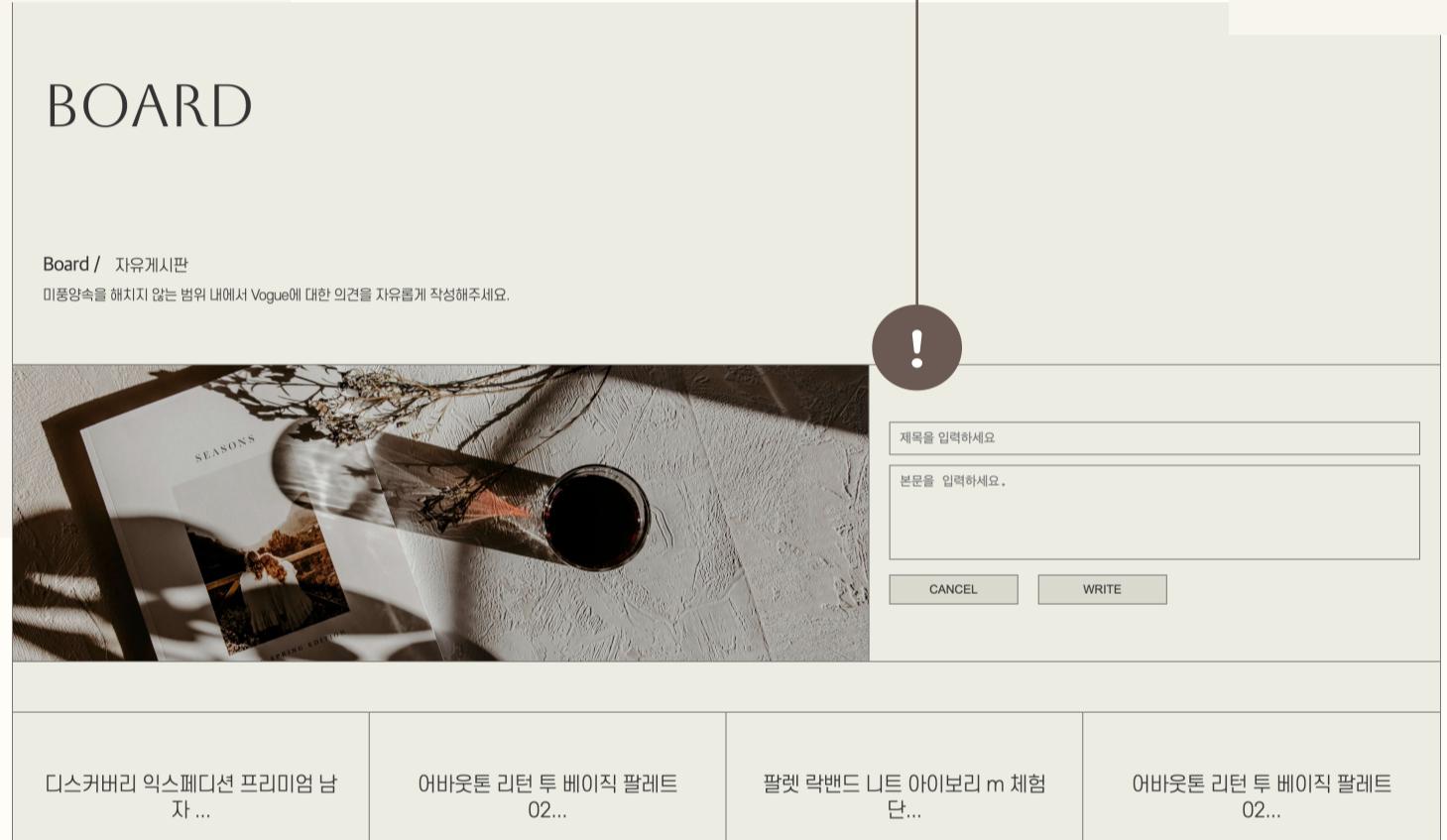
_board.scss

board component의 stylesheet

Board.js

자유게시판 페이지

- LocalStorage저장방식으로 CRUD 구현
- 데이터 전역 관리를 위해 main페이지 컴포넌트에서 관리
- 삭제 버튼 클릭 시 alert 창 호출
- 복수개 수정 방지 로직 구현



Create

입력한 값을 공백없이 저장해주는 함수
저장 완료 시 form 값을 다시 초기화

```
const createPost = () => {
  if (!input.current.value.trim() || !textarea.current.value.trim()) {
    resetForm();
    return alert('제목과 본문을 모두 입력하세요!');
  }
  setPosts([{ title: input.current.value, content: textarea.current.value }, ...Posts]);
  resetForm();
};
```

Get List

create 함수를 통해 localStorage에 저장된 값을 불러옴
LocalStorage에 데이터가 없으면 main-boardsection 컴포넌트에서 더미데이터로 데이터추가

```
const getLocalData = () => {
  const data = localStorage.getItem('post');
  return JSON.parse(data);
};
```

```
useEffect(() => {
  localStorage.setItem('post', JSON.stringify(Posts));
}, [Posts]);
```

boardMain 컴포넌트에서 getLocalData 함수를 만들어 dummy 데이터를 만들 어주고 로컬 데이터가 없으면 더미데이터를 반환하도록 구현하며 post 초기상태로 반환

Information from the React Portfolio

Board

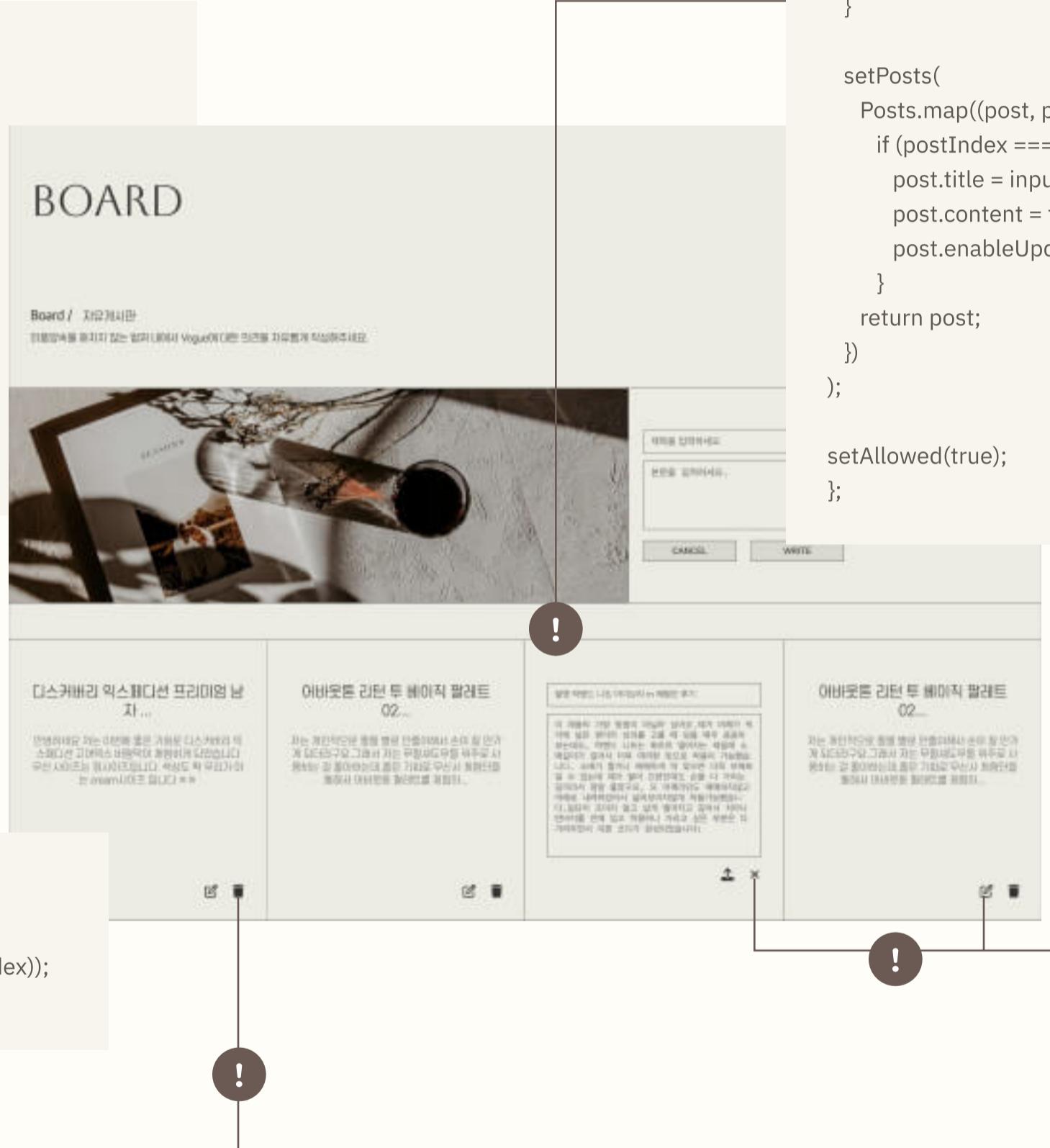
components
 └── sub
 └── Board.js
 board page에 대응하는 component

SCSS
 └── sub
 └── _board.scss
 board component의 stylesheet

Delete

게시글의 index 값으로 삭제게시글을 구분하고 해당 데이터를 호출

```
const deletePost = (delIndex) => {
  if (!window.confirm('해당 게시글을 삭제하겠습니까?')) return;
  setPosts(Posts.filter(_, postIndex => postIndex !== delIndex));
};
```



update

기존 post의 값 참조객체로 담아준 inputEdit값과 textAreaEdit 값으로 변경
게시글의 index값으로 수정게시글을 구분하며 해당 데이터를 호출

```
const updatePost = (updateIndex) => {
  if (!inputEdit.current.value.trim() || !textareaEdit.current.value.trim()) {
    return alert('수정할 제목과 본문을 모두 입력하세요.');
  }

  setPosts(
    Posts.map((post, postIndex) => {
      if (postIndex === updateIndex) {
        post.title = inputEdit.current.value;
        post.content = textareaEdit.current.value;
        post.enableUpdate = false;
      }
      return post;
    })
  );
  setAllowed(true);
};
```

수정이 활성화 될 시에 layout 변경

```
{post.enableUpdate ? 
  <>
  ...생략
</>
) : (
  <>
  ...생략
</>
)}
```

enable update

수정 버튼 활성화

```
const enableUpdate = (editIndex) => {
  if (!Allowed) return;
  setPosts(
    Posts.map((post, postIndex) => {
      if (postIndex === editIndex) post.enableUpdate = true;
      // post.enableUpdate 값을 false로 변경시 수정 취소 함수로 사용 가능
      return post;
    })
  );
  setAllowed(false);
```

Join

components

sub

Join.js

join page에 대응하는 component

SCSS

sub

_join.scss

join component의 stylesheet

Join.js

회원가입 페이지

- validation을 통한 input값 유효성 검사
- 성공 시 알림창 뜯 후 메인페이지로 반환
- title hover시 modal 출력



Hover Effect

title hover 시 pop up modal 출력

position값을 0px로 두었다가 hover 시 px값을 변경, opacity 값을 0에서 hover시 1로 변경하여 슬라이딩 효과를 줌

Validation - 폼 유효성 검사

빈값이거나 조건식에 부합하지 않으면 에러구문을 출력

type 별 대표값으로 예시 나머지 생략

```
const check = (value) => {
  const errs = {};
  const eng = /[a-zA-Z]/;
  const num = /[0-9]/;
  const spc = /[~!@#$%^&*])/;

  if (value.userid.length < 5) {
    errs.userid = '아이디를 5글자 이상 입력하세요';
  }
  if (value.pwd1.length < 5 || !eng.test(value.pwd1) || !num.test(value.pwd1) || !spc.test(value.pwd1)) {
    errs.pwd1 = '비밀번호는 5글자 이상, 영문, 숫자, 특수문자를 모두 포함하세요';
  }
  if (value.edu === '') {
    errs.edu = '최종 학력을 선택하세요.';
  }
  if (!value.gender) {
    errs.gender = '성별을 선택하세요.';
  }
  return errs;
};
```

에러 메세지 출력 화면

THE PERSON	
User ID	User ID 아이디를 5글자 이상 입력하세요
Password	Password 아이디를 입력하세요.
RePassword	RePassword 비밀번호는 5글자 이상, 영문, 숫자, 특수문자를 모두 포함하세요
Name	Name 비밀번호를 입력하세요.
Email	Email 이름을 두글자 이상 입력하세요
	Email 이메일은 8글자 이상 @를 포함하세요

Information from the React Portfolio

Contact

components

sub

Contact.js

contact page에 대응하는 component

SCSS

sub

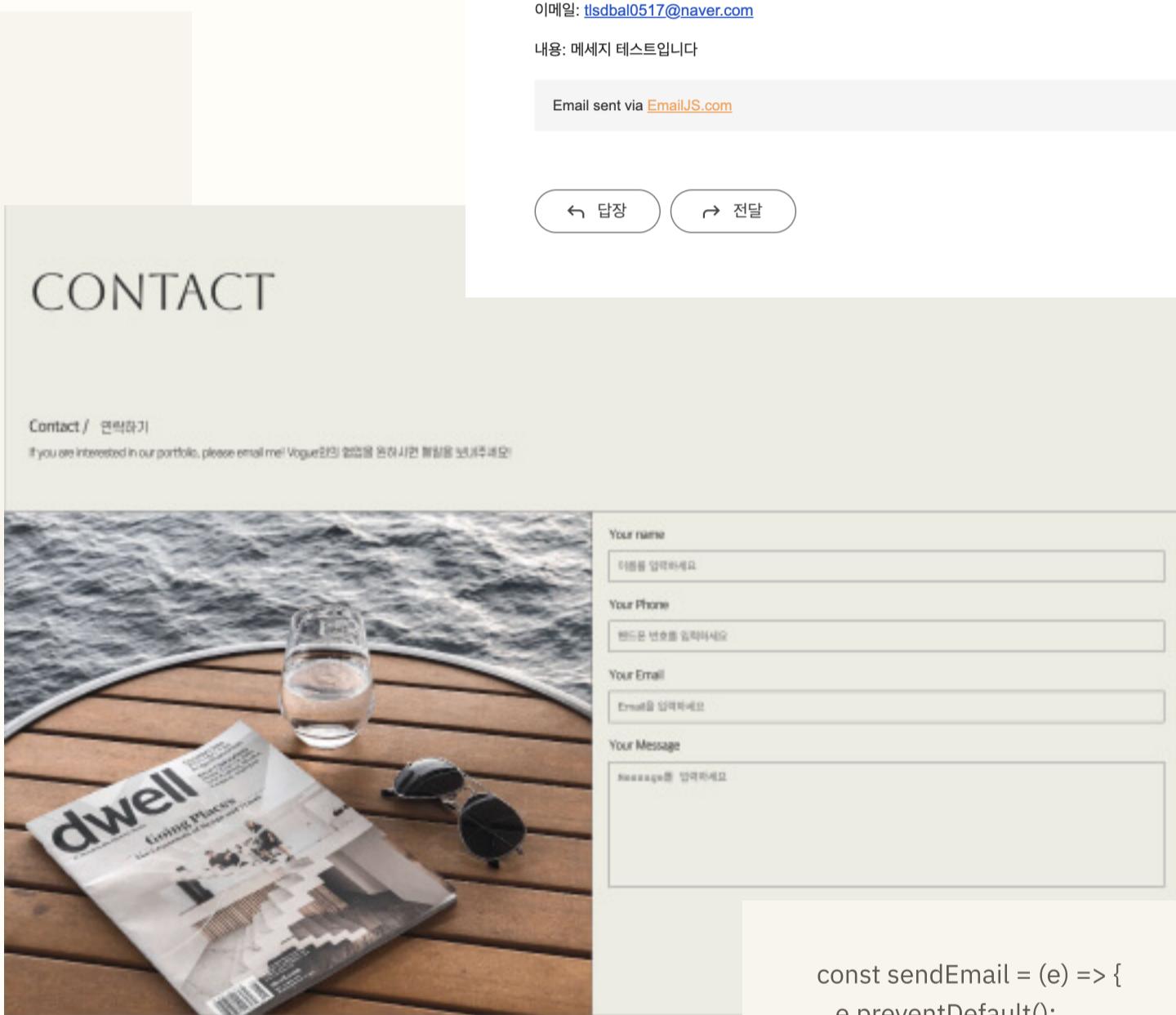
_contact.scss

contact component의 stylesheet

contact.js

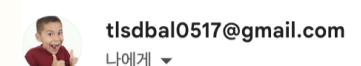
contact 이메일 연락 페이지

1. emailjs plugin을 사용한 메일 기능 구현
2. form validation으로 유효성 검사
3. 빈 값일 시 메일 보내지 않도록 방지 로직 구현



Success

전송 성공 시 받는 사람에게 출력되는 메일 화면



보낸사람: 신유미
연락처: 01012345678
이메일: tlsdbal0517@naver.com
내용: 메세지 테스트입니다

Email sent via [EmailJS.com](#)

↳ 담장 ↳ 전달

fail

빈 값을 넣을 시 유효성 검사에 의해 error 메세지 출력
join,js와 동일한 validation

Your name
이름을 입력하세요
이름을 2글자 이상 입력하세요

Your Phone
핸드폰 번호를 입력하세요
핸드폰 번호는 하이픈('-')없이 정확히 입력해주세요

Your Email
Email을 입력하세요
이메일은 8글자 이상 @를 포함하세요

Your Message
Message를 입력하세요
남기실 말씀을 적어주세요

Send

Send Email

validation으로 체크한 후 유효한 값이 들어갔을 경우 email.js plugin을 이용하여 결과값을 보내준 후 메일 보내기에 성공하면 form값을 리셋시켜준다

```
const sendEmail = (e) => {
  e.preventDefault();
  setErr(check(values));
  if (send) {
    emailjs
      .sendForm('service', 'template', form.current, 'id값')
      .then(
        (result) => {
          alert('전송되었습니다.');
        },
        (error) => {
          alert('전송을 실패했습니다.');
        }
      );
    form.current.reset();
  }
};
```

Alert

성공 시 alert창으로 전송되었습니다 모달 출력

