

코딩가이드 목차

1. 포트폴리오 설명

2. 공통 컴포넌트

2-1. Header, Footer

2-2. Layout

2-3. Menu, Popup, ScrollRe

3. Redux, Redux-saga

4. 메인 페이지

5. 서브 페이지

5-1. ABOUT

5-2. GALLERY

5-3. YOUTUBE

5-4. BLOG

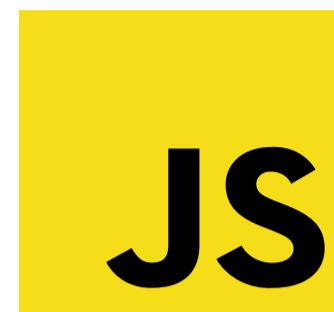
5-5. COMMUNITY

5-6. CONTACT

5-7. JOIN

6. 반응형 웹 레이아웃

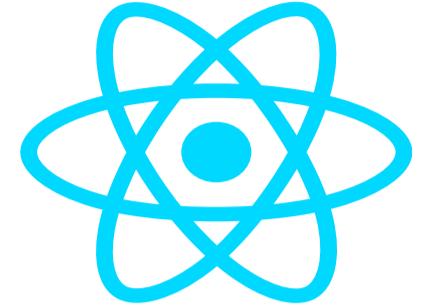
적용 스킬



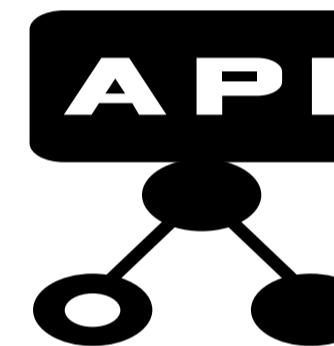
비구조화할당, 스프레드
연산자 등을 사용한 **바닐라**
라자바스크립트로 구현



코드의 중복사용이 적고
nesting이 간편하며, 변수
선언도 가능한 **scss** 사용



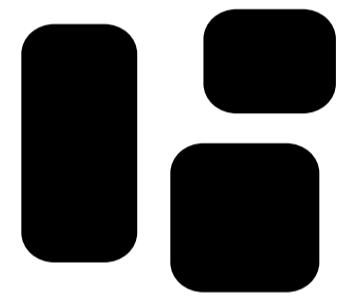
SPA 구조로 컴포넌트별
개발로 유지보수가 용이한
React 사용



youtube, flickr, kakao
map **API**를 활용하여 데
이터 활용



redux, **redux-saga**를
활용하여 state를 관리하
고 불필요한 코드의 중복
을 막음

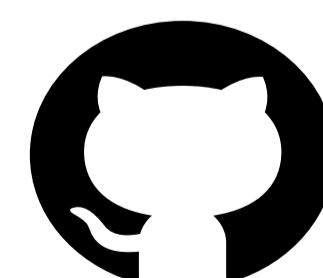


다양한 기기 환경에서 이
용할 수 있도록 **반응형 레
이아웃**으로 제작

개발 환경



Visual Studio
사용



github page를
통한 사이트 배포

1. 포트폴리오 설명

- 구조

 node_modules

 public

 DB member.js 데이터 파일 존재

 img 이미지들을 public 폴더에 넣어 절대경로로 활용

 index.html 하나의 html 파일로 많은 컴포넌트를 렌더링

 src

 class Anim.js을 포함.

 components common, main, sub 별로 컴포넌트 구성

 font 구글웹폰트가 아닌 파일로 된 폰트 파일들 존재

 redux redux 관련 파일들로 구성

 SCSS common, main, sub 별로 scss 파일 구성

 App.js

 index.js

JS App.js

```
return (
  <>
    <Switch>
      <Route exact path="/">
        <Header type={"main"} />
        <Main />
      </Route>
      <Route path="/">
        <Header type={"sub"} />
      </Route>
    </Switch>
    <Route path="/about" component={About}></Route>
    <Route path="/community" component={Community}></Route>
    <Route path="/gallery" component={Gallery}></Route>
    <Route path="/youtube" component={Youtube}></Route>
    <Route path="/blog" component={Blog}></Route>
    <Route path="/contact" component={Contact}></Route>
    <Route path="/join" component={Join}></Route>
    <Footer />
  </>
);
```



.gitignore

node_modules는 용량이 크기 때문에 git push 할 때 제외하고 업로드



package.json

npm install 시에 --save 옵션을 통해 package.json에 기록하고 git에서 해당 프로젝트 다운로드 시 프로젝트에서 사용한 모듈을 npm install로 간단히 설치할 수 있다

2. 공통 컴포넌트

2-1. Header, Footer

<Main Header> props값에 따라 각각의 디자인 적용



<Sub Header>



<Footer>



Redux를 사용한 상태관리

```
const Members = useSelector((state) => state.memberReducer.members);
```

2. 공통 컴포넌트

2-2. Layout



The screenshot shows a web page with a large, semi-transparent 'A' logo in the background. In the foreground, there is a sub-page titled 'About Creative All Things'. The page has a light beige background with a blue border around the main content area. At the top right, there is a 'Beige' logo and a 'JOIN' button. Below the title, there is some placeholder text: 'Lorem, ipsum.'

Code Snippet:

```
<> <figure className={`${sub_visual ${props.name}_sub`}>
  <div className="wrap">
    <span>${props.name.substr(0, 1)}</span>
    <div className="txt">
      <h2>${props.name}</h2>
      <p>Creative All Things</p>
      <div className="cursor" ref={cursor}></div>
    </div>
    <p>Lorem, ipsum.</p>
  </div>
</figure>
<section className={`${content_section ${props.name}`} ${props.children}>
</section>
</>
```

Annotations and Explanations:

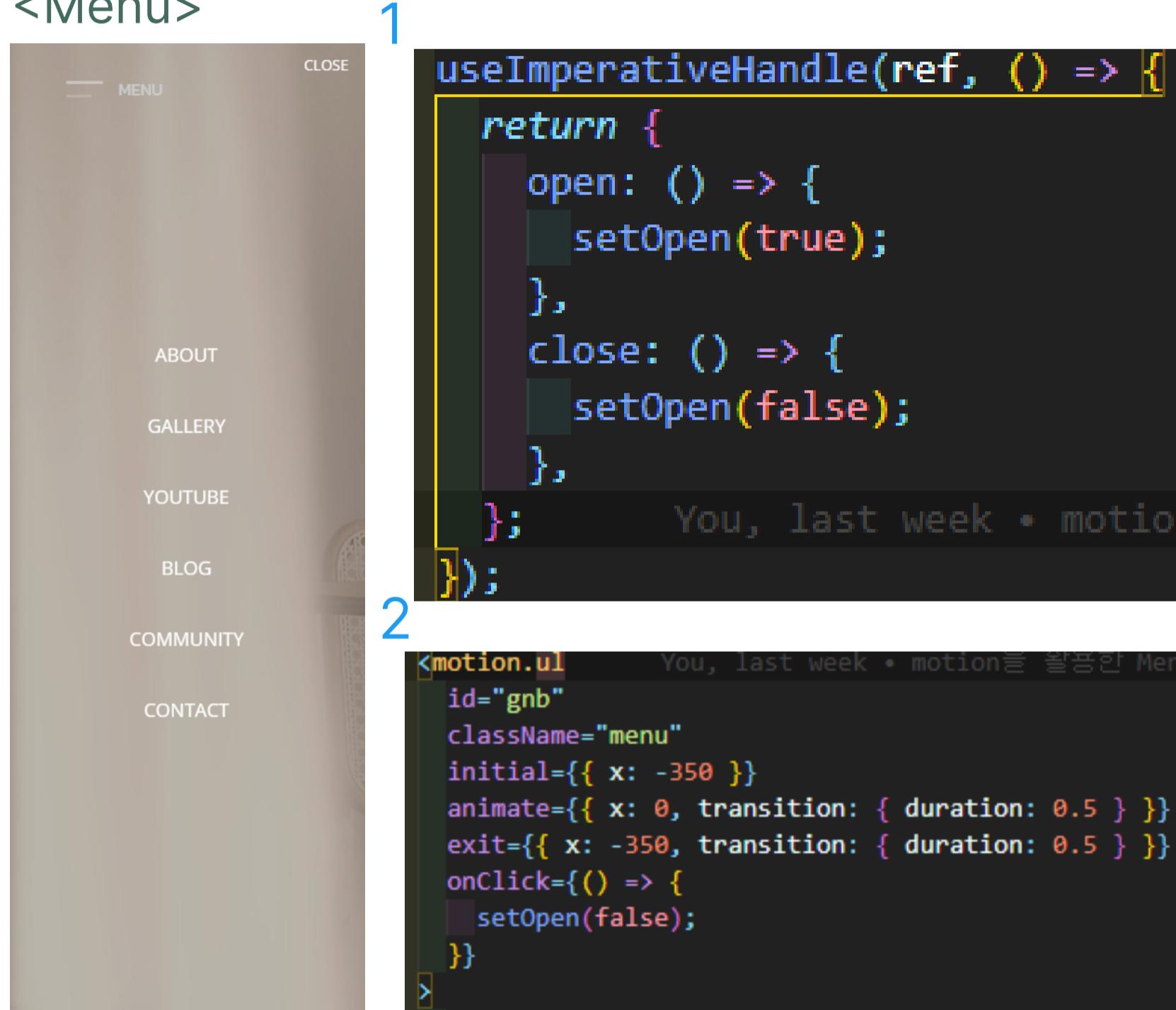
- An arrow points from the first highlighted line to the explanatory text: "props.name 값을 className 으로 설정해 각 sub page 의 backgroundColor 값을 다르게 설정".
- An arrow points from the second highlighted line to the explanatory text: "props.name의 앞글자를 가져와 뒤의 배경글자처럼 효과 구현".
- An arrow points from the third highlighted line to the explanatory text: "원형 모양의 cursor pointer 삽입, h2 태그를 만나면 확대되면서 색상 반전 효과 적용".
- A blue arrow points down from the bottom of the code snippet to the explanatory text: "sub 컴포넌트들을 Layout 컴포넌트로 묶어 코드의 중복을 막고 마운트 시에 컴포넌트가 아래에서 위로 등장하는 모션 추가".
- A blue arrow points down from the explanatory text to the bottom of the page.



2. 공통 컴포넌트

2-3. Menu, Popup, Scrollre

<Menu>



1 forwardRef, useImperativeHandle을 통해 Header 컴포넌트에서 컴포넌트를 참조할 수 있게 해줌.

상위 컴포넌트에서 Menu 컴포넌트의 함수를 사용할 수 있음

2 framer 라이브러리를 사용해 버튼 클릭 시 position 값을 변경해서 왼쪽에서 오른쪽으로 등장하는 모션 적용

<Popup>

```
<motion.aside
  className="popup"
  initial={{ opacity: 0 }}
  animate={{ opacity: 1, transition: { duration: 0.5 } }}
  exit={{ opacity: 0, transition: { duration: 0.5 } }}>
<motion.div
  className="con"
  initial={{ opacity: 0 }}
  animate={{
    opacity: 1,
    transition: { duration: 0.5 },
  }}
  exit={{ opacity: 0, transition: { duration: 0.5 } }}>
  {props.children}
</motion.div>
</motion.aside>
```

Menu 컴포넌트와 동일한 방법으로
GALLERY, YOUTUBE,
BLOG 서브페이지에서 이미
지 클릭 시 팝업창 생성

<Scrollre>

```
const { pathname } = useLocation();
useEffect(() => {
  window.scrollTo(0, 0);
}, [pathname]);

return null;
}
```

<Link>로 페이지 이동 시에 스크롤 위치가 그대로 이동하는 문제점 발생
→ useLocation Hook으로 비구조화 할당으로 pathname값을 받아 pathname값이 바뀔 때마다 스크롤이 0부터 시작하도록 설정

3. Redux, Redux-saga

- 구조



Redux



actionType.js



api.js



reducers.js



saga.js



store.js

```
const sagaMiddleware = createSagaMiddleware();

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

applyMiddleware함수를 통해 saga를 걸쳤다가 실행.

```
useEffect(() => {
  dispatch({ type: types.MEMBER.start });
  dispatch({ type: types.YOUTUBE.start });
  dispatch({ type: types.FLICKR.start, opt: { type: "interest" } });
  dispatch({ type: types.GALLERY.start });
}, []);
```

App.js에서 dispatch를 통해 start type을 호출

```
export const getMember = async () => {
  const url = path + `/DB/member.json`;
  return await axios.get(url);
};
```

getMember뿐만 아니라 getFlickr, getYoutube 등 App.js에서 axios로 데이터를 받아오는 구문들이 많아져 App 컴포넌트가 복잡해지는 것을 방지하기 위해 api.js에 모아 놓음.
유지 보수하는데 있어서 편리하다.

```
export const MEMBER = {
  start: "MEMBER_START",
  success: "MEMBER_SUCCESS",
  error: "MEMBER_ERROR",
};
```

```
export const YOUTUBE = {
  start: "YOUTUBE_START",
  success: "YOUTUBE_SUCCESS",
  error: "YOUTUBE_ERROR",
};
export const FLICKR = {
  start: "FLICKR_START",
  success: "FLICKR_SUCCESS",
  error: "FLICKR_ERROR",
};
```

```
const memberReducer = (state = {
  members: [] }, action) => {
  switch (action.type) {
    case types.MEMBER.start:
      return { ...state };
    case types.MEMBER.success:
      return { ...state, members: action.payload };
    case types.MEMBER.error:
      return { ...state, error: action.payload };
    default:
      return state;
  }
};
```

memberReducer, youtubeReducer, flickrReducer등 action.type값을 받아 값에 따라 상태를 관리함

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

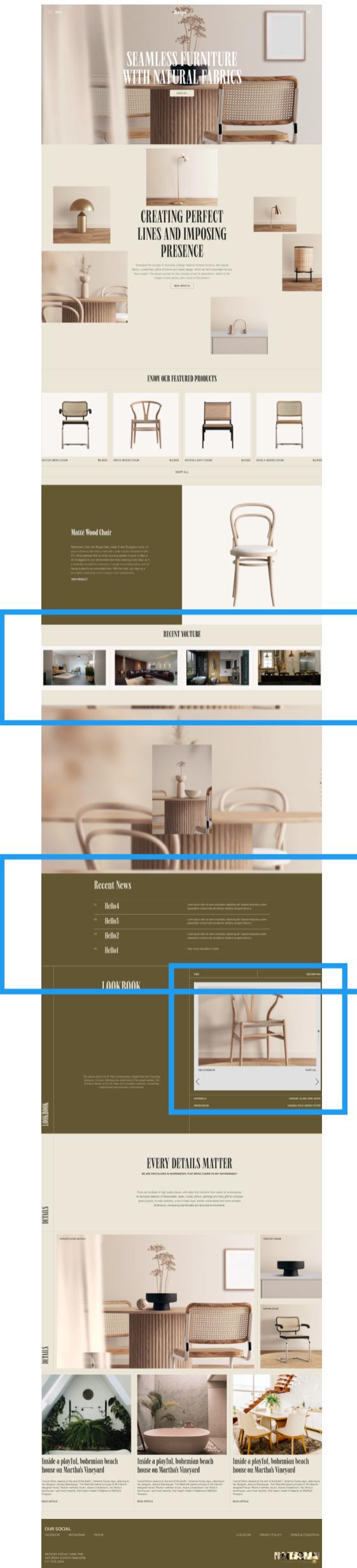
export function* callMember() {
  yield takeLatest(types.MEMBER.start,
  returnMember);
}

export function* returnMember() {
  try {
    const response = yield call(getMember);
    yield put({ type: types.MEMBER.success, payload:
    response.data.data });
  } catch (err) {
    yield put({ type: types.MEMBER.error, payload: err
  });
  }
}
```

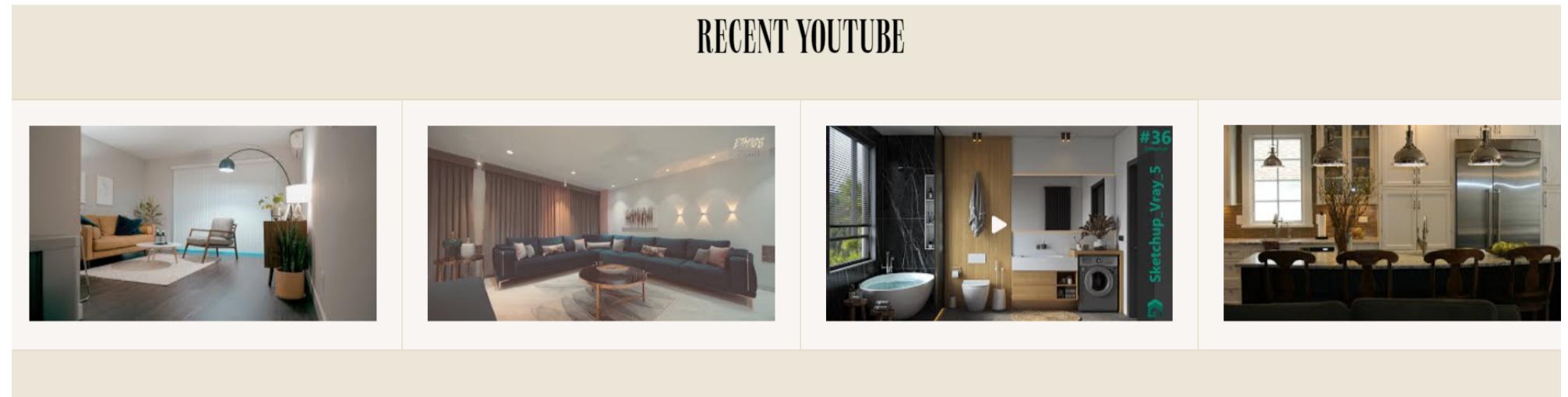
4. 메인페이지

• 구조

<Visual/>
<Section1 />
<Section2 />
<RecentYoutube /> 1
<Visual2 />
<News /> 2
<Section3 /> 3
<Section4 />
<Section5 />
<ScrollBtns /> 4
<ScrollTopBtn /> 5

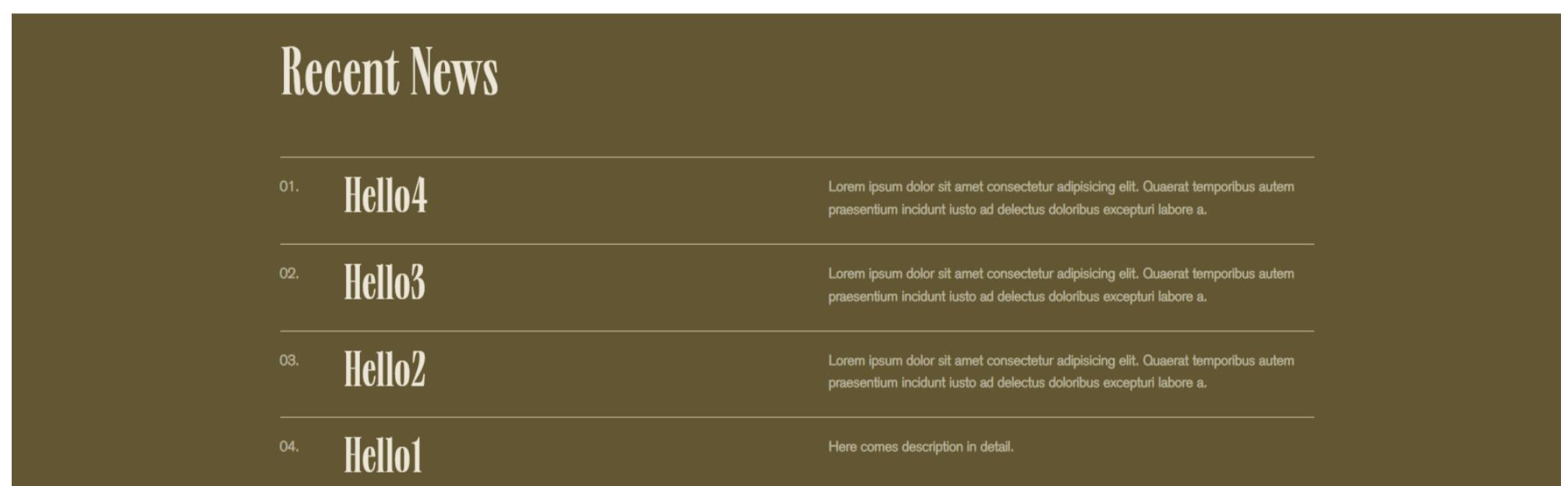


1 const youtubeData = useSelector((state) => state.youtubeReducer.youtube);

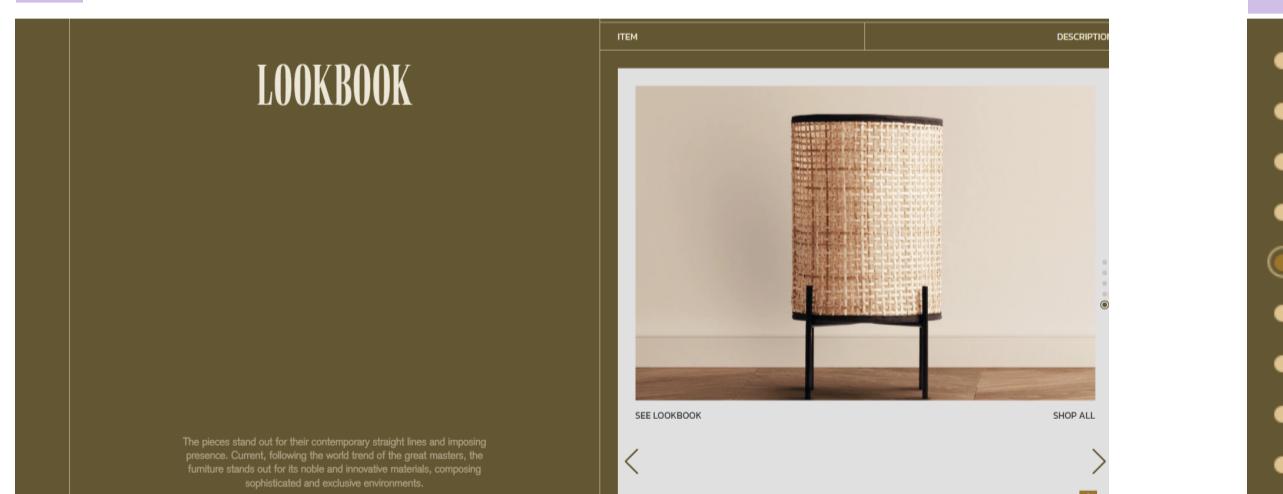


Redux로 관리된 스테이트값을 가져와서 4개만 화면에 나타남.

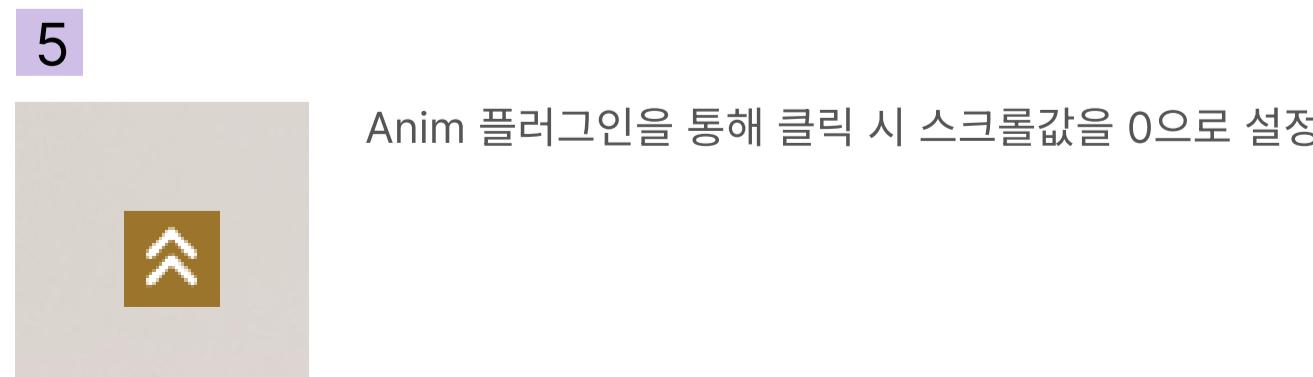
2 localStorage.setItem을 통해 Community 컴포넌트에서 CRUD로 구현한 데이터를 Main페이지에 불러옴



3 Swiper를 통해 슬라이드 기능 구현



4 각 섹션의 offsetTop값을 구하여 스크롤의 값이 해당 섹션의 offsetTop을 넘는 순간 섹션의 index값을 이용해 버튼의 활성화를 제어하고, 클릭 시 Anim 플러그인을 사용해서 해당 섹션의 offsetTop으로 scroll값으로 위치시킴.



5 Anim 플러그인을 통해 클릭 시 스크롤값을 0으로 설정

5. 서브페이지

5-1. ABOUT

public/DB/member.json JSON 파일로 데이터 관리

```
"data": [  
  { "name": "Matt Ryan", "position": "Vice President",  
  "pic": "member1.jpg" },  
  {  
    "name": "Julia Jessica",  
    "position": "Stocker",  
    "pic": "member2.jpg"  
},
```



```
const Members = useSelector((state) => state.memberReducer.members);
```

Redux에서 데이터를 불러옴

이미지 호버 효과

1. transition-delay 값을 통해 각각의 박스들이 순서대로 대각선에서 올라오는 효과 적용.
2. 이미지 Scale, filter: Sepia 값 변경.

5. 서브페이지

5-2. GALLERY

```
const getItem = () => {
  axios.get(url).then((json) => {
    const data = json.data.photos.photo;
    setItems(data);
    setPopLoading(true);
    setTimeout(() => {
      frame.current.classList.add("on");
      setLoading(false);
    }, 1000);
  });
};
```

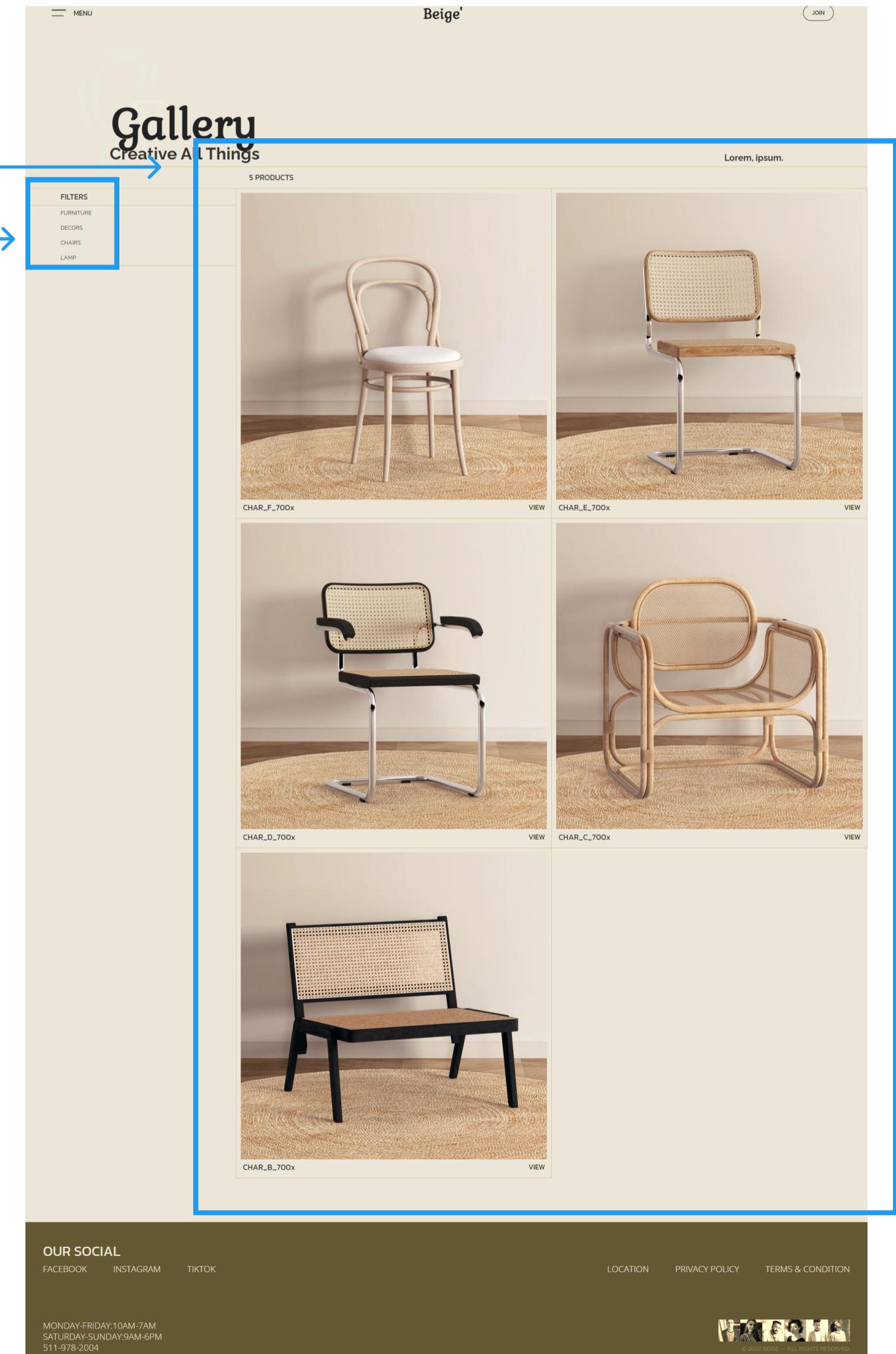
axios로 데이터 호출
Loading 값에 따라 loading 이미지의 display 값 설정
class "on"이 붙으면 아래에서 등장하는 효과 적용

```
const getFilter = (title) => {
  axios.get(url).then((json) => {
    const data = json.data.photos.photo;
    const regex = new RegExp(title);
    const result = data.filter((item) => regex.test(item.title));
    setItems(result);
    setTimeout(() => {
      frame.current.classList.add("on");
      setLoading(false);
    }, 1000);
  });
};
```

data 객체 값 중 title 키 값에 따라 이미지를 분류하기 위해 정규표현식을 활용해 인자로 받은 값을 filter를 통해
인자로 받은 타이틀 값과 이미지의 타이틀 값이 같은 이미지만 분류

```
onClick={() => {
  setLoading(true);
  frame.current.classList.remove("on");
  getFilter("DECOR");
}}
```

li마다 onClick 이벤트를 통해 getFilter 함수를 호출
인자로 받은 타이틀값과 맞는 이미지를 출력



5. 서브페이지

5-3. YOUTUBE

```
const youtubeData = useSelector((state) => state.youtubeReducer.youtube);
```

Redux에 저장된 youtube 데이터를 호출

```
style={{ transform: `rotate(${(360 / youtubeData.length) * idx }deg) translateY(-155%)` }}  
youtubeData를 map함수로 반복시키고 원형 모양으로 article을 퍼뜨림.
```

버튼 클릭시 index값을 변경하여 활성화된 article은 class “on”을 붙이고 비활성화된 article은 opacity 값을 낮춰 투명하게 만듬.



5. 서브페이지

5-4. BLOG



January 8, 2022 0 Comments

PITT flickr 데이터의 title값을 동적으로 생성

Lorem ipsum dolor sit amet consectetur adipisicing elit.
 Cupiditate commodi laborum vero nesciunt? Voluptate
 expedita illum delectus, officia alias voluptatem vel dolorem
 optio, vero, dolores nostrum amet nobis sint. Consectetur,
 reiciendis modi dignissimos saepe animi molestiae
 asperiores eius at mollitia.

 138308002@N07
Photographer

flickr 데이터의 이미지를 올린 오너의 아이디값을 동적으로 생성

```
const [Month, setMonth] = useState([
  "January",
  "February",
  "March",
  "April",
  "May",
  "June",
  "July",
  "August",
  "September",
  "October",
  "November",
  "December",
]);
```

```
let monthRandom = parseInt(Math.random() * 12);
let dayRandom = parseInt(Math.random() * 31);
```

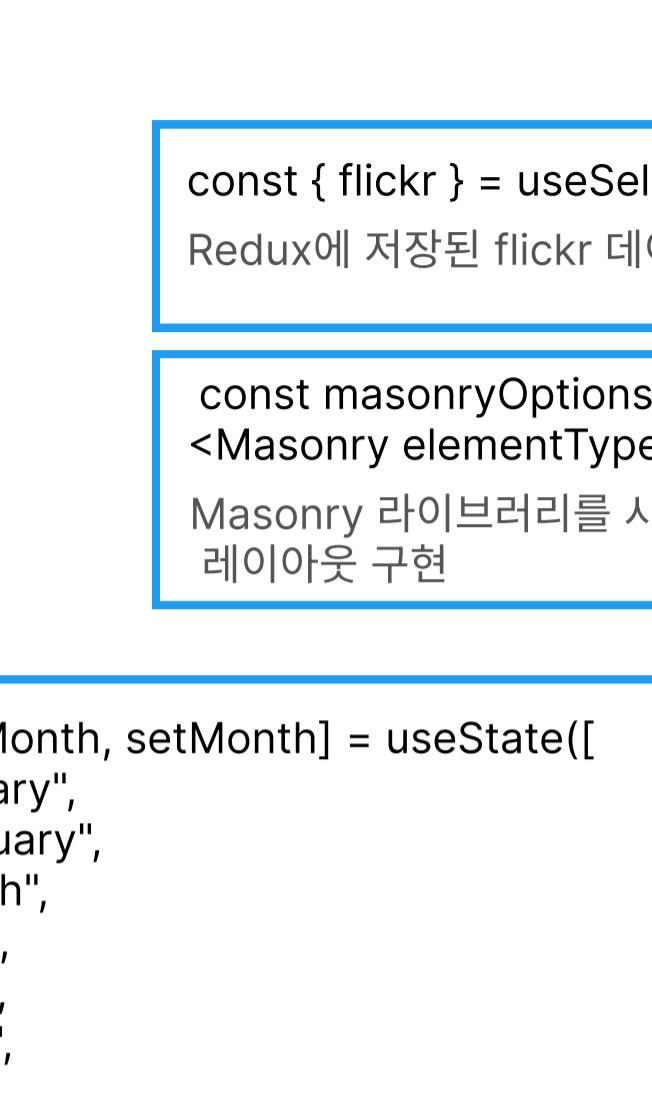
random 함수를 사용해 Month 배열의 인덱스값을 받아 동적으로 생성하고, day값도 1~31까지 랜덤으로 생성

```
const { flickr } = useSelector(state) => state.flickrReducer);
Redux에 저장된 flickr 데이터를 호출
```

```
const masonryOptions = {transitionDuration: "0.5s"};
<Masonry elementType={"div"} options={masonryOptions}>
Masonry 라이브러리를 사용해 높이 값이 달라도 빈곳을 채워주는
레이아웃 구현
```

```
dispatch({ type: "FLICKR_START", opt });
if (opt.type === "search") {
  url = `https://www.flickr.com/services/rest/?method=${method2}&per_page=${num}&api_key=${key}&format=json&nojsoncallback=1&tags=${opt.tags}`;
}
```

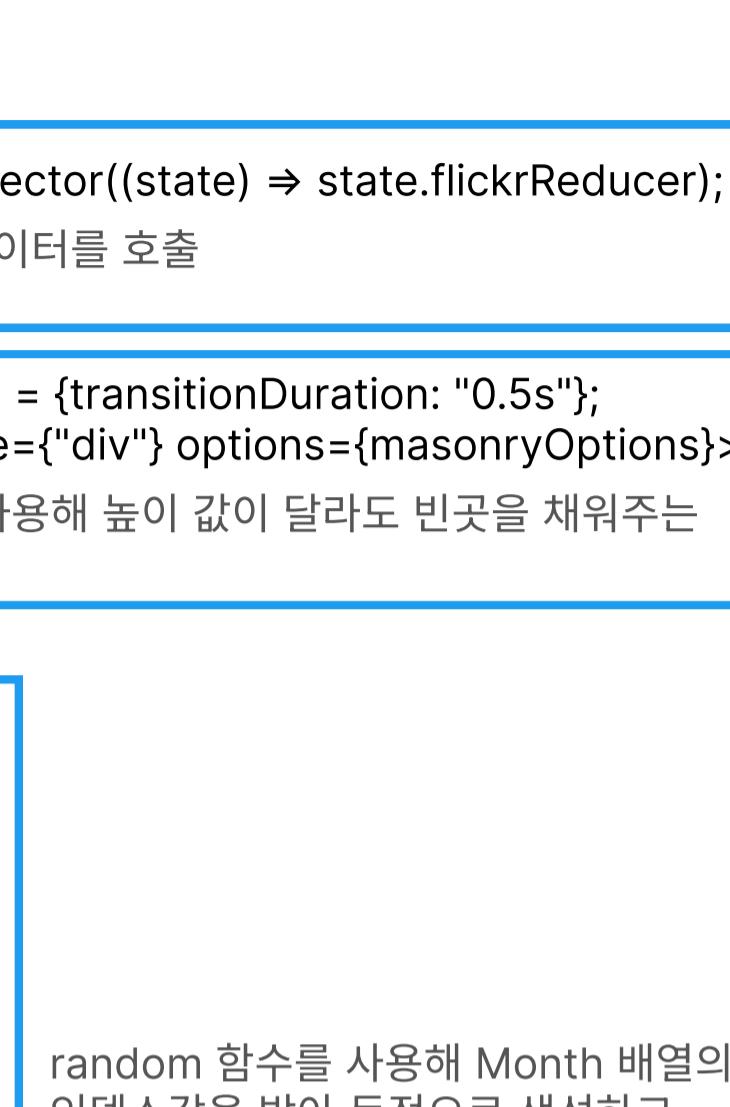
input의 value 값을 넣어서 opt의 인자로 넣고 tag값을 사용해 flickr api를 호출



January 8, 2022 0 Comments

PITT flickr 데이터의 title값을 동적으로 생성

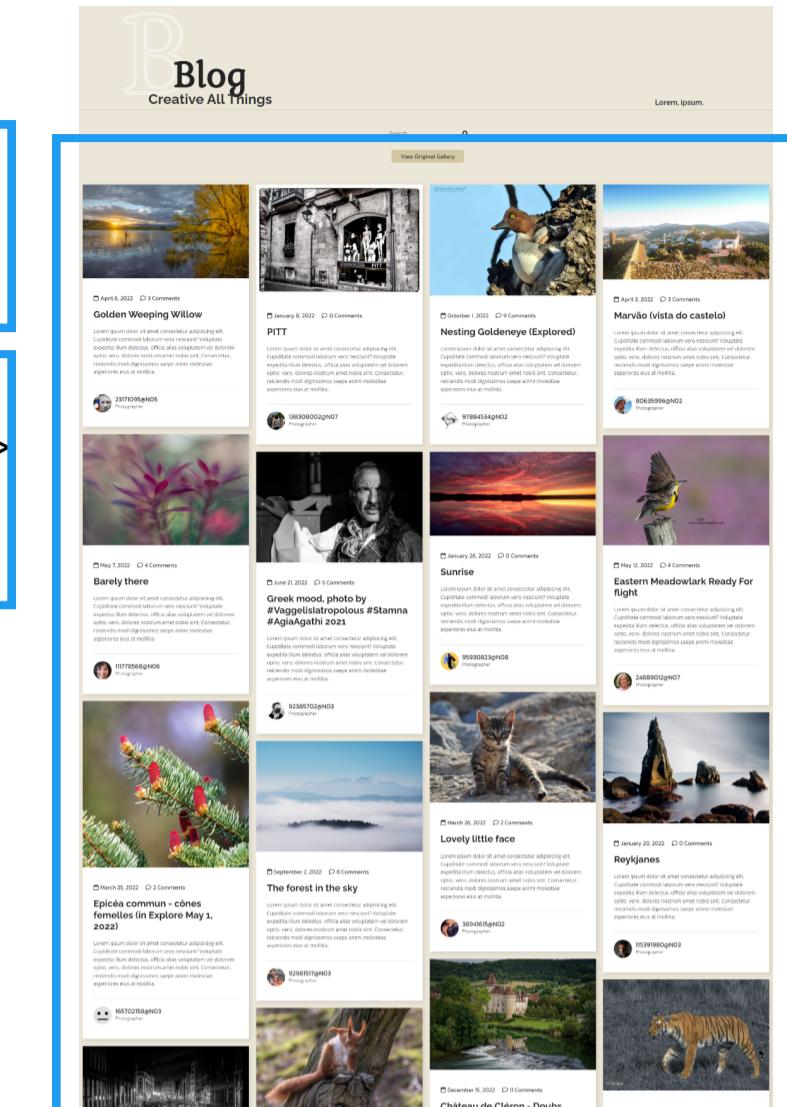
Lorem ipsum dolor sit amet consectetur adipisicing elit.
 Cupiditate commodi laborum vero nesciunt? Voluptate
 expedita illum delectus, officia alias voluptatem vel dolorem
 optio, vero, dolores nostrum amet nobis sint. Consectetur,
 reiciendis modi dignissimos saepe animi molestiae
 asperiores eius at mollitia.



const [Month, setMonth] = useState([
 "January",
 "February",
 "March",
 "April",
 "May",
 "June",
 "July",
 "August",
 "September",
 "October",
 "November",
 "December",
]);

let monthRandom = parseInt(Math.random() * 12);
let dayRandom = parseInt(Math.random() * 31);

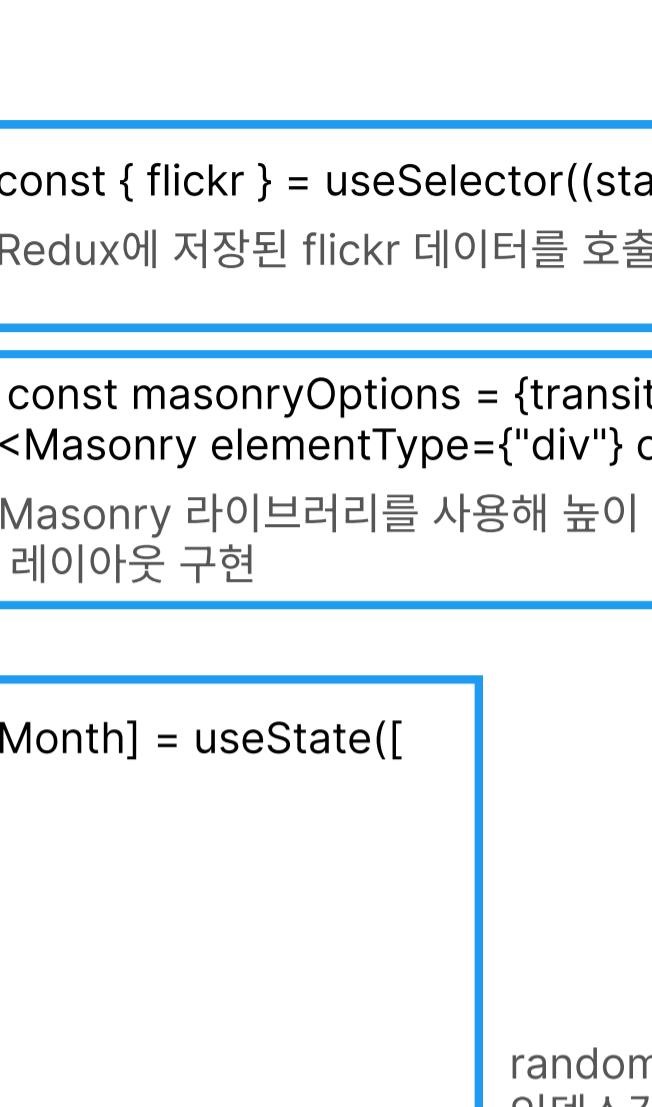
random 함수를 사용해 Month 배열의 인덱스값을 받아 동적으로 생성하고, day값도 1~31까지 랜덤으로 생성



const { flickr } = useSelector(state) => state.flickrReducer);
Redux에 저장된 flickr 데이터를 호출



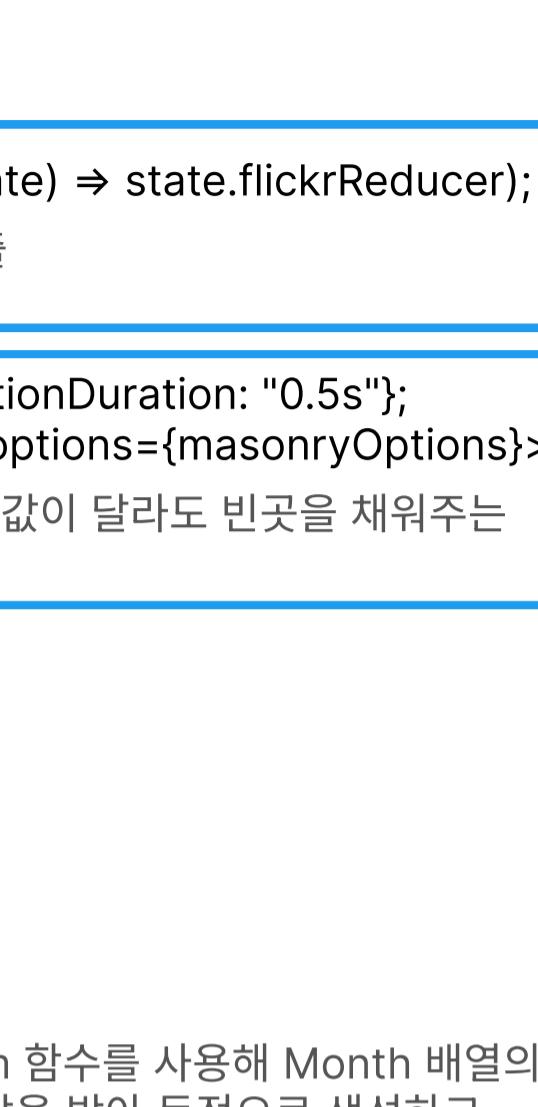
const masonryOptions = {transitionDuration: "0.5s"};
<Masonry elementType={"div"} options={masonryOptions}>
Masonry 라이브러리를 사용해 높이 값이 달라도 빈곳을 채워주는
레이아웃 구현



const [Month, setMonth] = useState([
 "January",
 "February",
 "March",
 "April",
 "May",
 "June",
 "July",
 "August",
 "September",
 "October",
 "November",
 "December",
]);

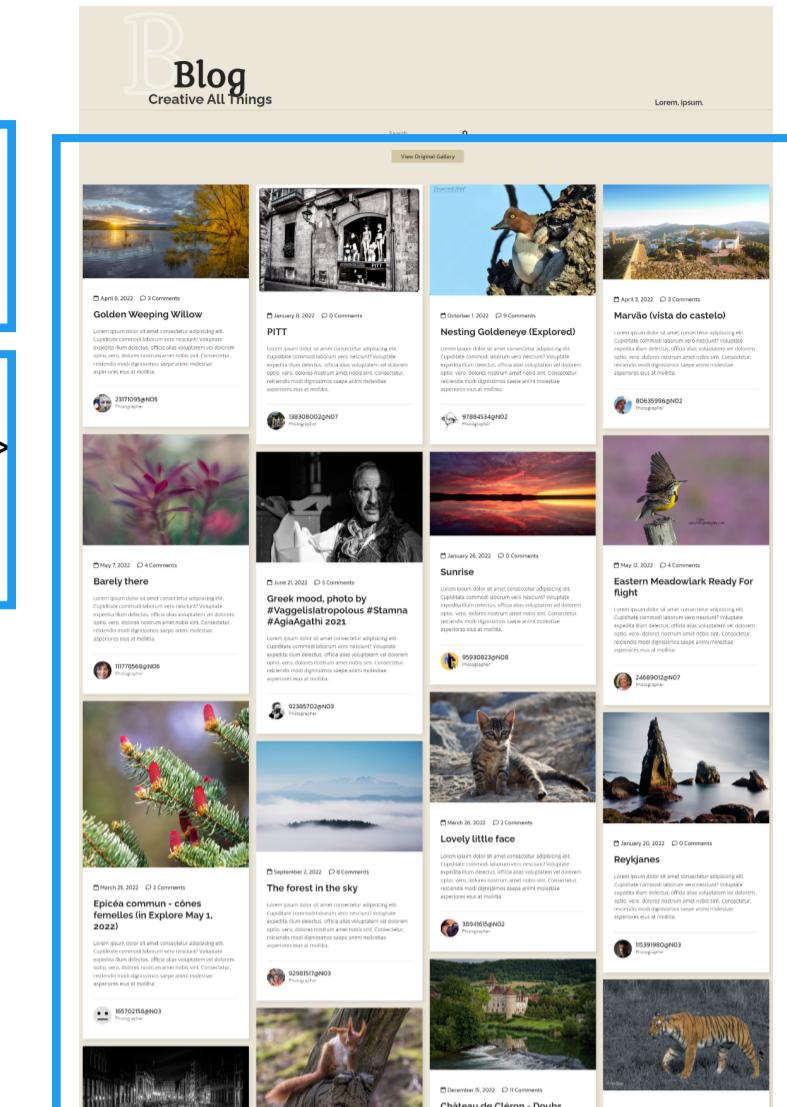
let monthRandom = parseInt(Math.random() * 12);
let dayRandom = parseInt(Math.random() * 31);

random 함수를 사용해 Month 배열의 인덱스값을 받아 동적으로 생성하고, day값도 1~31까지 랜덤으로 생성



dispatch({ type: "FLICKR_START", opt });
if (opt.type === "search") {
 url = `https://www.flickr.com/services/rest/?method=\${method2}&per_page=\${num}&api_key=\${key}&format=json&nojsoncallback=1&tags=\${opt.tags}`;
}

input의 value 값을 넣어서 opt의 인자로 넣고 tag값을 사용해 flickr api를 호출



const { flickr } = useSelector(state) => state.flickrReducer);
Redux에 저장된 flickr 데이터를 호출



const masonryOptions = {transitionDuration: "0.5s"};
<Masonry elementType={"div"} options={masonryOptions}>
Masonry 라이브러리를 사용해 높이 값이 달라도 빈곳을 채워주는
레이아웃 구현

5. 서브페이지

5-5. COMMUNITY CRUD 구현

```
const getLocalData = () => {
  let data = localStorage.getItem("posts");
  return JSON.parse(data);
};

const [posts, setPosts] = useState(getLocalData());
localStorage.getItem으로 데이터를 posts에 저장
```

```
const createPost = () => {
  const inputVal = input.current.value.trim();
  const textareaVal = textarea.current.value.trim();
  if (!inputVal || !textareaVal) {
    alert("제목과 본문을 입력해주세요");
    return;
  }
  setPosts([{ title: inputVal, content: textareaVal }, ...posts]);
  resetPost();
};
```

Hello4
Lorem ipsum dolor sit amet consectetur adipisicing elit. Quaerat temporibus autem praesentium incidentum iusto ad delectus doloribus excepturi labore a.

```
const editPost = (index) => {
  setPosts(
    posts.map((post, idx) => {
      if (idx === index) post.enableUpdate = true;
      return post;
    })
  );
};
```

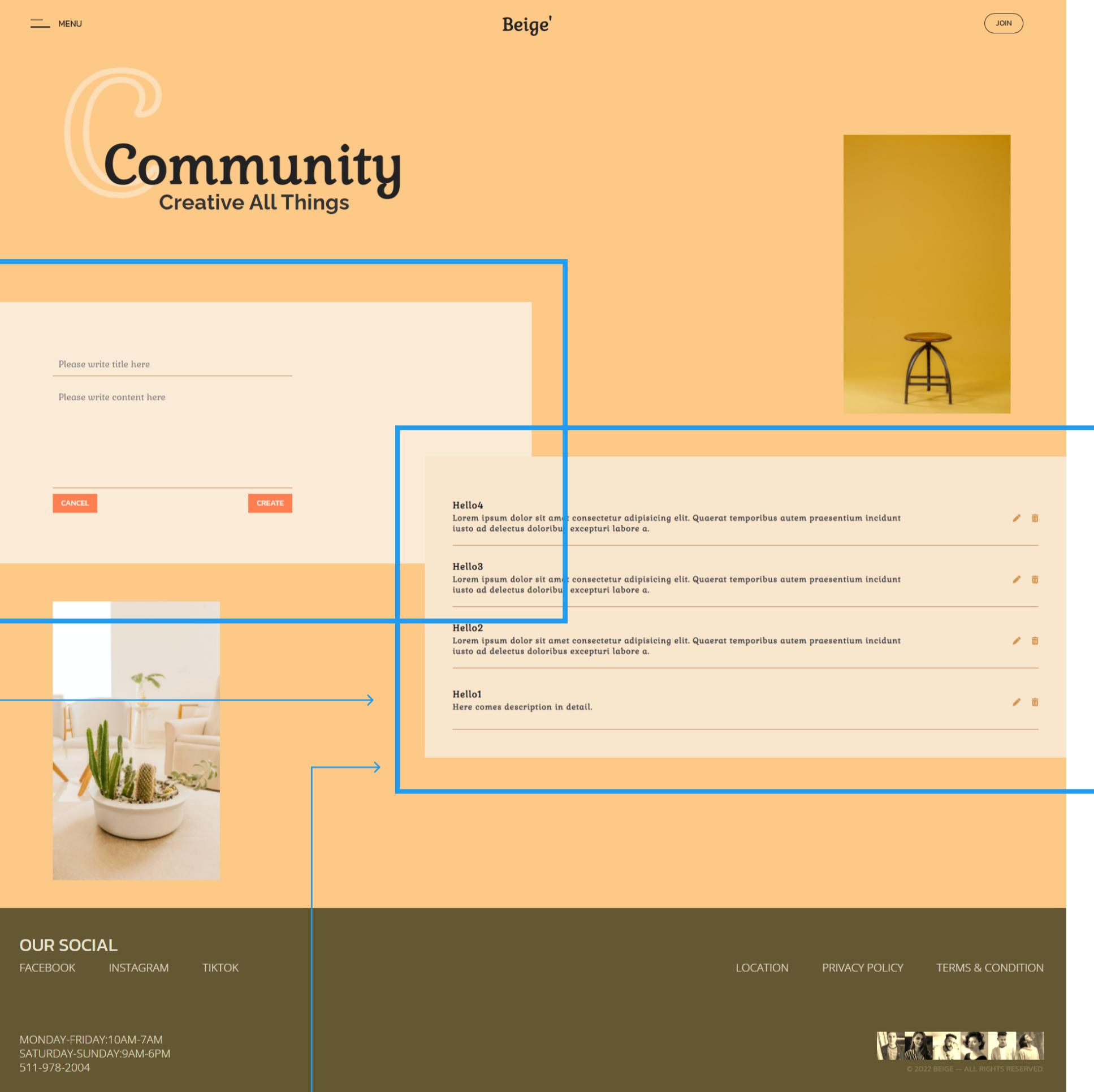
```
const updatePost = (index) => {
  const inputVal = editInput.current.value.trim();
  const textareaVal = editTextarea.current.value.trim();
  if (!inputVal || !textareaVal) {
    alert("제목과 본문을 입력해주세요");
    return;
  }
};
```

```
setPosts(
  posts.map((post, idx) => {
    if (idx === index) {
      post.title = editInput.current.value;
      post.content = editTextarea.current.value;
      post.enableUpdate = false;
    }
    return post;
  })
);
```

CREATE

input, textarea 값을 받아 title, content 값으로 posts 스테이트에 저장.

if문으로 값이 없을 경우 예외처리.



UPDATE

trim으로 공백 예외처리 후 index값을 비교하여 index값이 같을 경우 input, textarea의 value를 다시 posts 스테이트에 저장

```
{posts.map((post, idx) => {
```

map함수로 반복을 돌면서 출력

```
const deletePost = (index) => {
  console.log(index);
  setPosts(posts.filter((_, idx) =>
    idx !== index));
};
```

DELETE

filter함수로 index값을 비교하여 index값이 다른 경우만 출력

5. 서브페이지

5-6. CONTACT

1. kakao API 호출
2. 화면 리사이즈 시, 마커 가운데 위치
3. 교통정보 on, off 기능
4. 지점 클릭 시 지도 이동
5. 지도 타입 변경, 확대, 축소 기능
6. 마커 이미지 수정

```
const info = [
  {
    title: "SEOUL",
    latlng: new kakao.maps.LatLng(37.554740104452726, 126.9705978395484),
    imgSrc: `${path}/img/marker1.webp`,
    imgSize: new kakao.maps.Size(100, 100),
    imgPos: { offset: new kakao.maps.Point(50, 50) },
  },
]
const [mapInfo, setmapInfo] = useState(info);
```

```
useEffect(()=>{
  container.current.innerHTML = ""; 지도가 재호출될때마다 겹치는 것을 방지
  const mapOption = {
    center: mapInfo[Index].latlng,
    level: 3,
  };

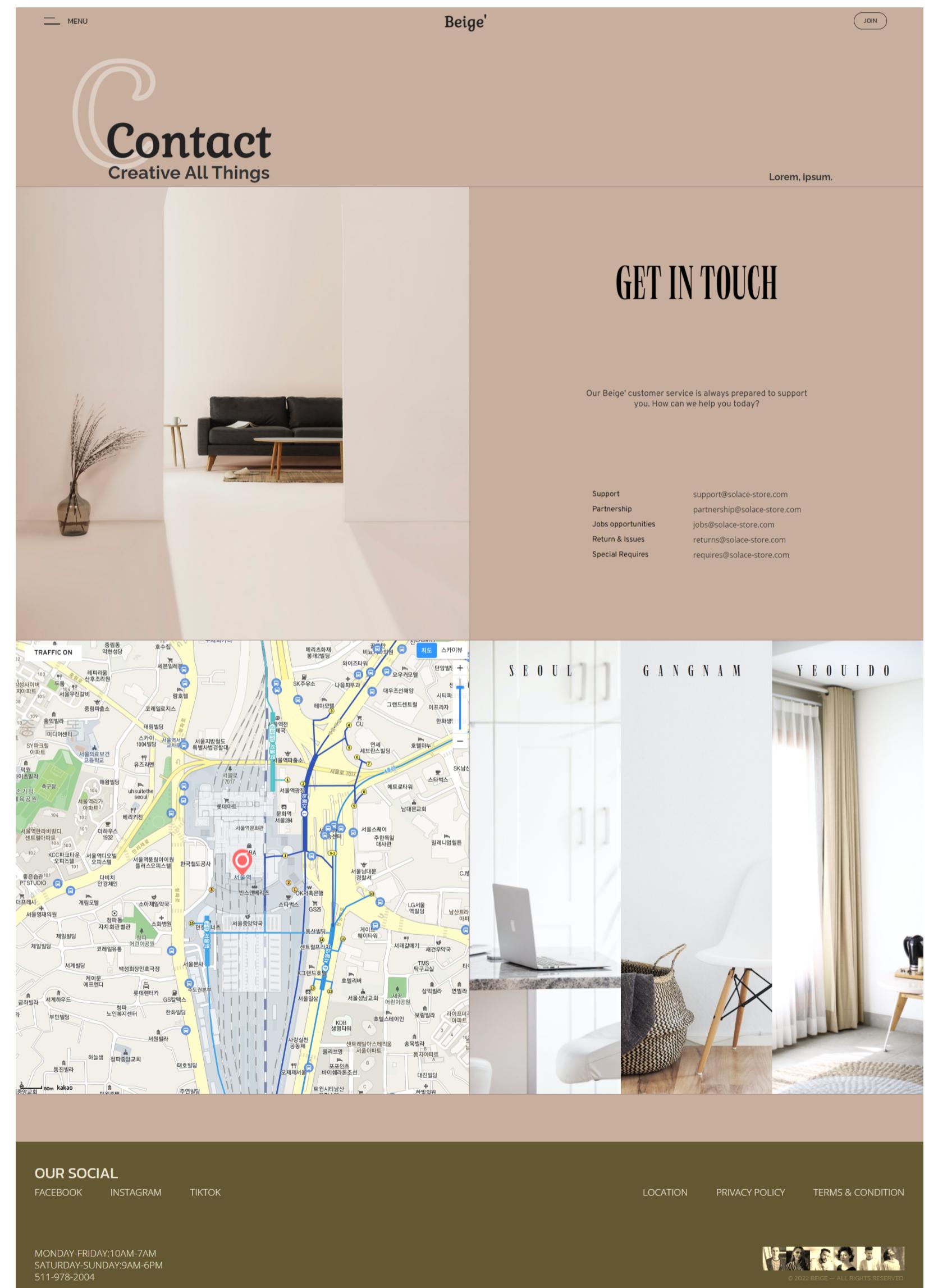
  const mapInstance = new kakao.maps.Map(container.current, mapOption);
  setMap(mapInstance);
})
```

```
const mapInit = () => {
  mapInstance.setCenter(mapInfo[Index].latlng);
};
window.addEventListener("resize", mapInit);
```

info에 map 정보값을 저장 후
mapInfo 스테이트에 저장

mapInfo값을 토대로 초기 맵
호출

윈도우 창 사이즈가 변경될때마다 지도의 위치가 달라지는 문제를
해결하기 위해 resize 될 때마다 지도의 위치가 center로 오는 함
수 호출



The screenshot shows the Beige' Contact page. At the top right, there are 'MENU', 'Beige', and 'JOIN' buttons. Below the header, the word 'Contact' is prominently displayed with the tagline 'Creative All Things'. A large image of a modern interior room with a sofa and a vase is centered. To the right, a section titled 'GET IN TOUCH' contains a message: 'Our Beige' customer service is always prepared to support you. How can we help you today?'. Below this, there are links for Support, Partnership, Jobs opportunities, Return & Issues, and Special Requires, each with an email address. The main content area features a map of Seoul's Gangnam and Yeouido districts, showing various landmarks like Lotte World Tower, SK Tower, and the Han River. To the right of the map, there are three smaller images: a laptop on a desk, a chair under a window, and a close-up of a lamp. At the bottom, there are social media links for Facebook, Instagram, and TikTok, along with a 'LOCATION', 'PRIVACY POLICY', and 'TERMS & CONDITION' link. The footer also includes a copyright notice: '© 2022 BEIGE - ALL RIGHTS RESERVED'.

5. 서브페이지

5-7. JOIN

```

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

  if (Val.userid.length < 5) {
    errs.userid = "아이디를 5글자 이상 입력하세요";
  }
  if (
    Val.pwd1 < 5 ||
    !eng.test(Val.pwd1) ||
    !num.test(Val.pwd1) ||
    !spc.test(Val.pwd1)
  ) {
    errs.pwd1 = "비밀번호는 5글자 이상 특수문자 영어를 포함해주세요";
  }
  if (Val.pwd1 !== Val.pwd2 || !Val.pwd2) {
    errs.pwd2 = "비밀번호가 일치하지 않습니다.";
  }
  if (Val.email < 5 || !/@/.test(Val.email)) {
    errs.email = "@를 포함해 5글자 이상 입력해주세요";
  }
  if (Val.comments.length < 10) {
    errs.comments = "10글자 이상 남겨주세요";
  }
  if (!Val.gender) {
    errs.gender = "성별을 선택해주세요";
  }
  if (!Val.interests) {
    errs.interests = "관심사를 하나 이상 선택해주세요";
  }
  if (Val.edu === "") {
    errs.edu = "최종 학력을 선택해주세요";
  }
  return errs;
};

```

if문으로 Val 값들마다 예외 처리 후 send 버튼에 클릭 이벤트로 check함수가 실행.

```

const handleRadio = (e) => {
  const { name } = e.target;
  const isCheck = e.target.checked;
  setVal({ ...Val, [name]: isCheck });
};

const handleCheck = (e) => {
  let isCheck = false;
  const { name } = e.target;
  const inputs = e.target.parentElement.querySelectorAll("input");
  inputs.forEach((el) => {
    if (el.checked) isCheck = true;
  });
  setVal({ ...Val, [name]: isCheck });
};

const handleSelect = (e) => {
  const { name } = e.target;
  const isSelected = e.target.options[e.target.selectedIndex].value;
  setVal({ ...Val, [name]: isSelected });
};

```

check 여부를 확인하고 check된 값을 Val에 저장.
select도 마찬가지로 selected된 index값을 Val에 저장

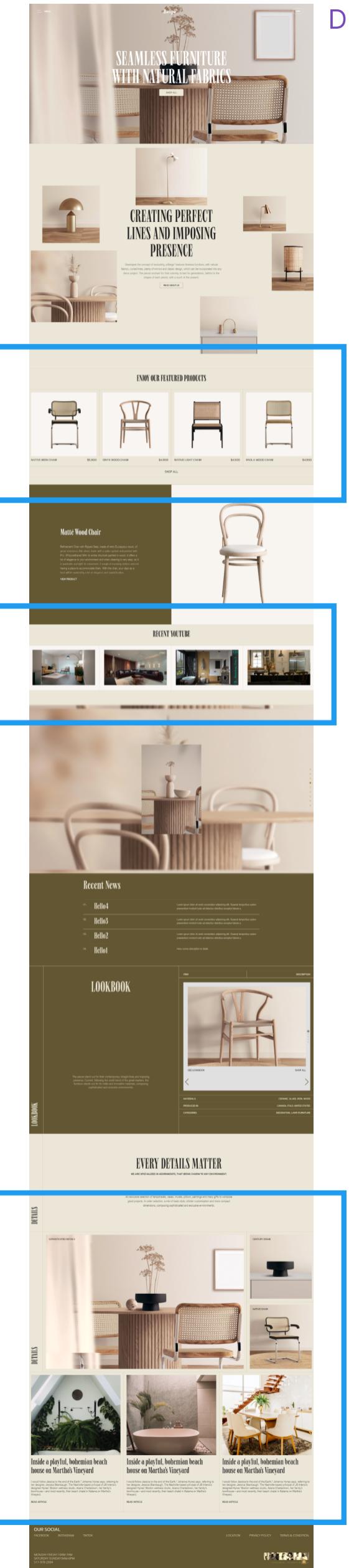
6. 반응형 웹 레이아웃

SCSS 변수 처리

```
$tablet: 1179px;  
  
$mobile: 539px;  
  
@media screen and (max-width: $tablet) {}  
@media screen and (max-width: $mobile) {}
```

해당 브라우저 폭이 1179px 미만일 경우 tablet scss 적용

해당 브라우저 폭이 539px 미만일 경우 mobile scss 적용



DESKTOP 환경

TABLET 환경

MOBILE 환경

이미지 사이즈 및
레이아웃 변화

이미지 사이즈 및
레이아웃 변화

이미지 사이즈 및
레이아웃 변화