EURA's

PROTFOLIO GUIDE

#0. PROJECT LIST .

PROJECT 1.

massive .

PROJECT 2.

Hello & Co

PROJECT 3.

Eura’s notice

#0_1 CREATE A PLUGIN

- 재사용되는 스크립트를 클래스화 시켜 플러그인으로 사용하므로써 개발 시간 단축

src/class

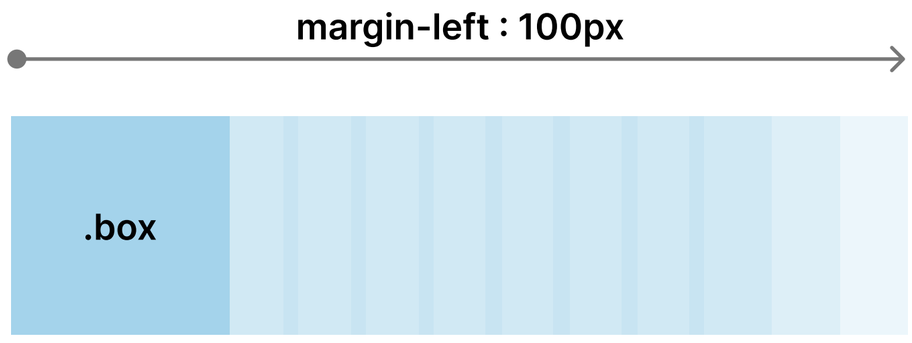
- anime.js 객체 및 스크롤 애니메이션을 위한 플러그인
- banner.js 버튼 롤링 배너 플러그인

How to use

```
new Anime('.box', {
  prop: 'margin-left',
  value: 100,
  duration: 500,
  callback: ()⇒{
  }
});
```

instance를 통해 불러온 class안에 1번째 인자값으로 선택자 전달
2번째 인자값으로 객체를 전달해 상세 속성설정
prop : 애니메이션이 적용될 속성
value : 변경될 속성값
duration : 진행시간
callback : 해당 인스턴스가 종료 후 실행될 함수

```
class Anime {
 constructor(selector, option) {
  this.selector = document.querySelector(selector);
  this.option = option;
    ⋮
  requestAnimationFrame((time) ⇒ { this.run(time) });
 }

 run(time){
  실행할 코드 작성
    ⋮
 }
}
```



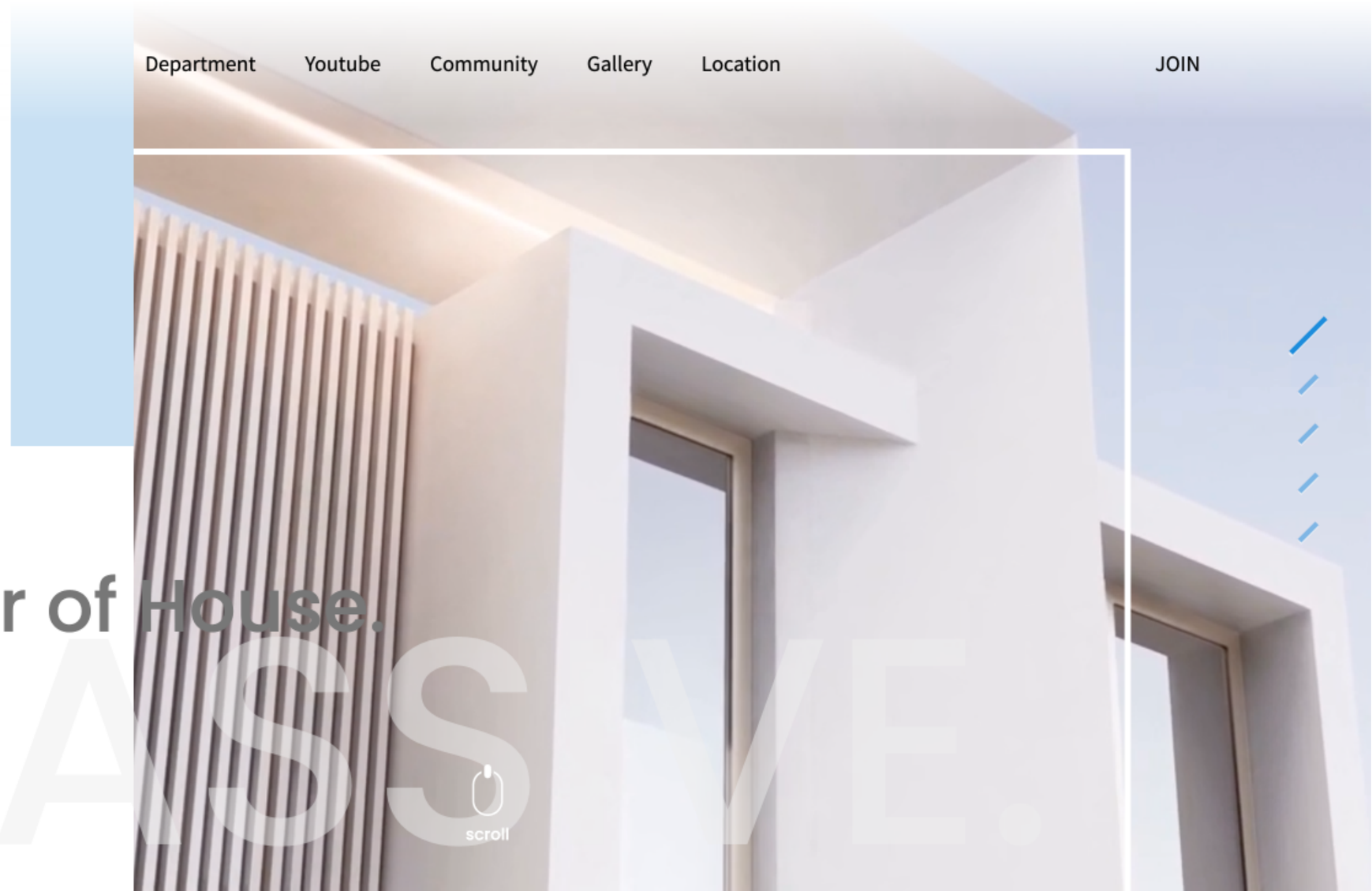
#1. INFO .

NAME : MASSIVE.

URL : https://eurako.github.io/massive/

SPEC :

RESPONSIVE : PC, TABLET, MOBILE



#1_1. CONCEPT

- 건축 concept으로 깨끗하고 각진 느낌을 살린 선과 면을 활용한 레이아웃을 구성

#FFFFFF	#1D8DDC	#161B7F	#777777	#333333

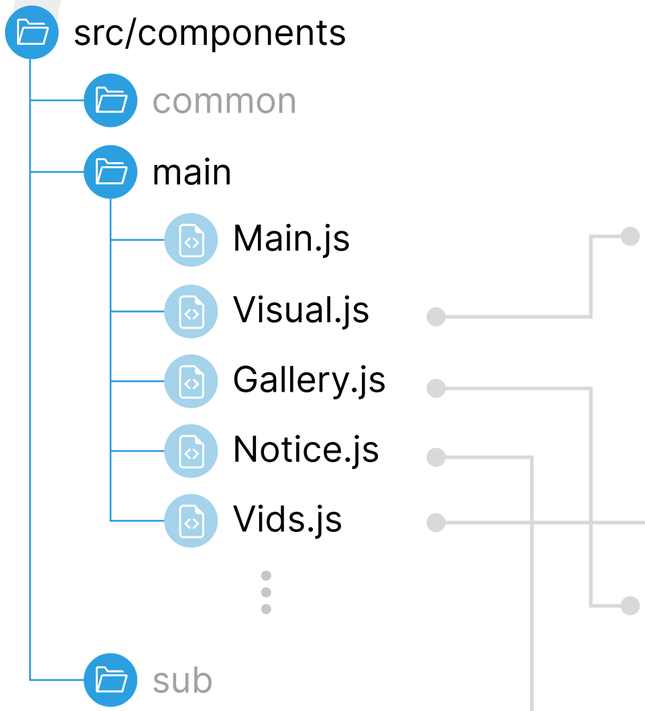
#1_2. STRUCTURE

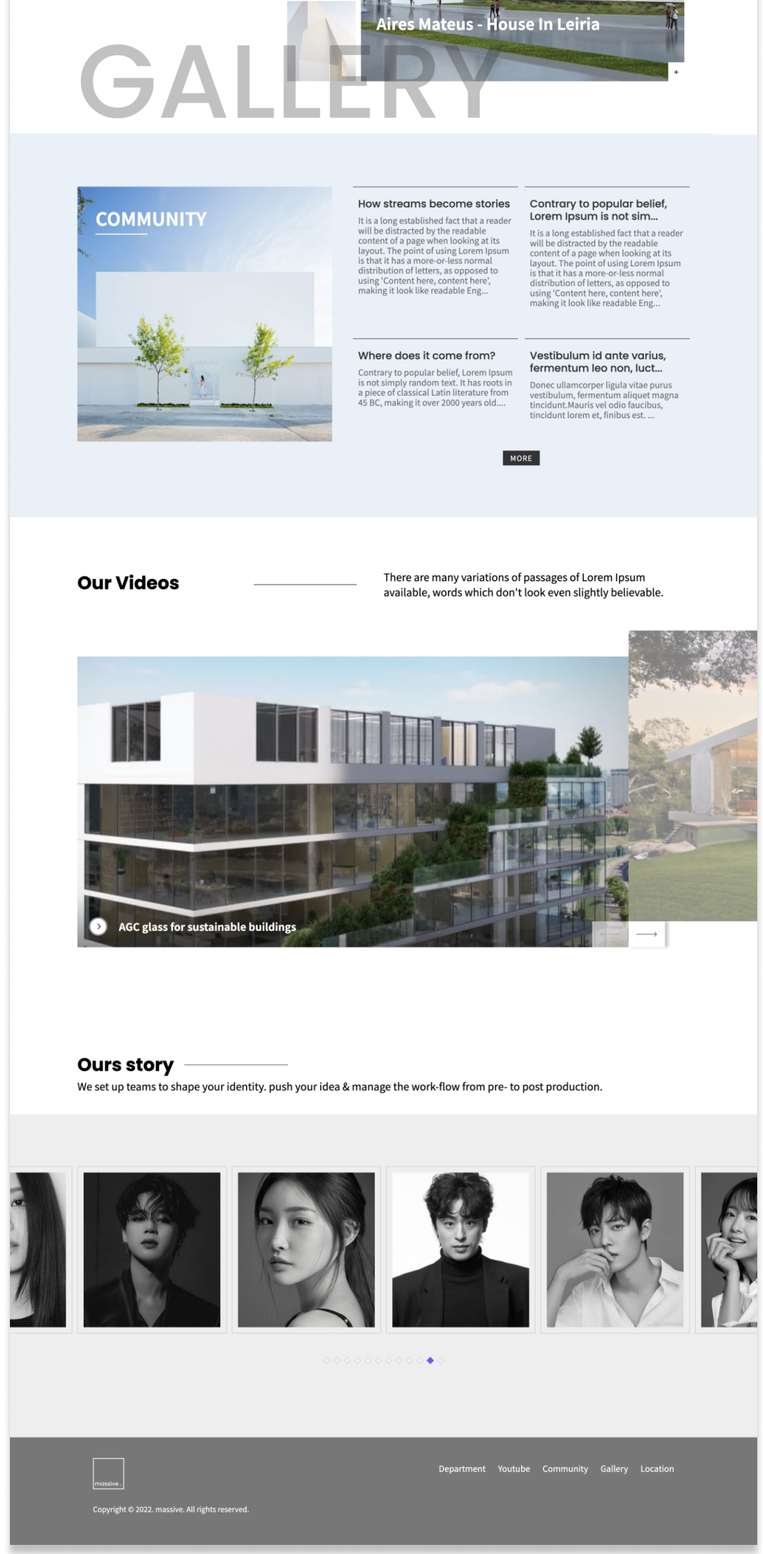
- 중복되는 컨텐츠를 효율적으로 관리하기 위해 컴포넌트기반의 구조를 가지고 있는 React와 Sass를 활용하여 프로젝트를 구성

ROOT	
node_modules	
public	
img	환경변수를 사용하여 효율적인 작업을 위해 public폴더 안에서 관리
index.html	최종 랜더링 결과 화면
src	
class	ES6문법 class문으로 개발한 plugin 폴더
components	view 영역
redux	react의 단점인 단방향 데이터를 보완하기 위한 redux saga 폴더
scss	css 전처리기
App.js	초기 component
index.js	components를 조합하여 랜더링하는 파일
pakage.json	
.gitignore	

#1_3. MAIN STRUCTURE

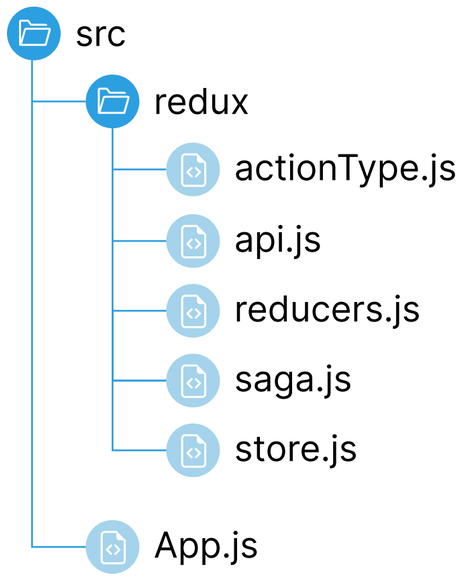
- section별로 컴포넌트를 분리하여 코드의 가독성을 높이고 유지보수에 유리
- 스크롤애니메이션을 처리하기 위해 Main.js 컴포넌트화





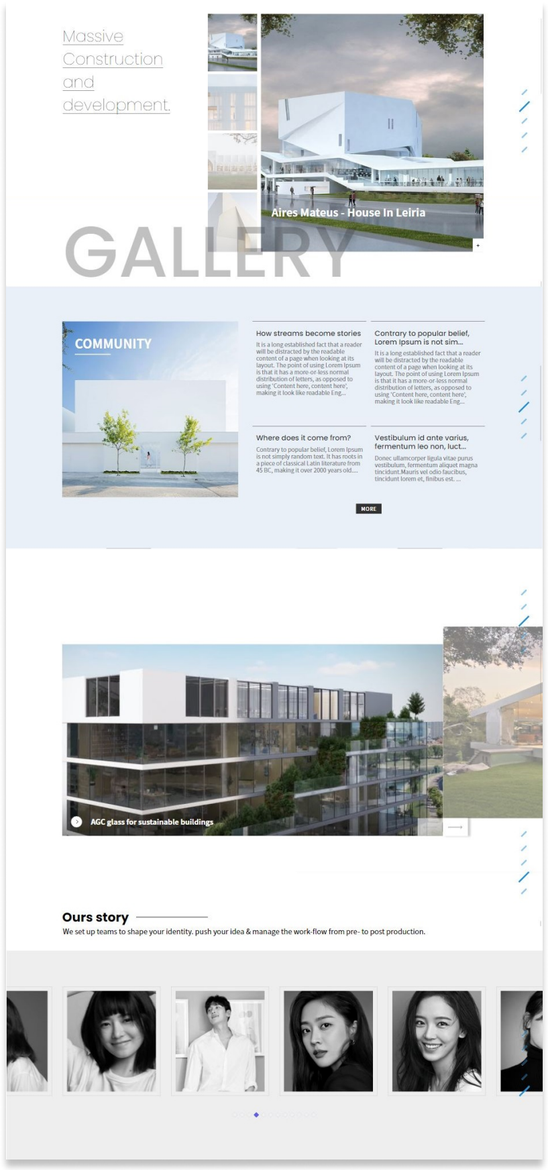
#1_4 STATE MANAGE MENTE TOOLS

- 중복으로 사용되는 데이터를 redux와 reduxt-saga를 적용하여 데이터를 외부에서 관리
- **dispatch** Hook을 사용하여 최상단 컴포넌트인 App.js에서 데이터 불러오기

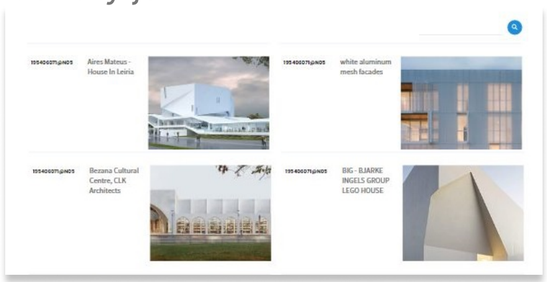



```
import * as types from './redux/actionType';

useEffect(() => {
  dispatch({ type: types.MEMBERS.start });
  dispatch({ type: types.YOUTUBE.start });
  dispatch({
   type: types.GALLERY.start,
   opt: { type: 'user', count: 20},
  });
 }, []);
```

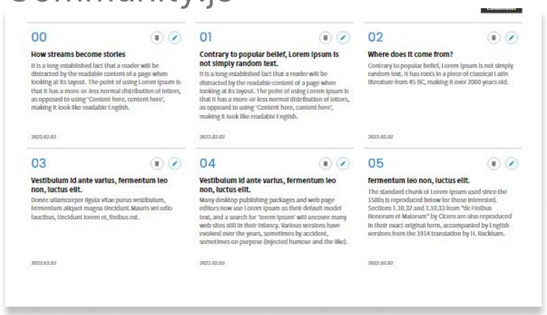
Main.js



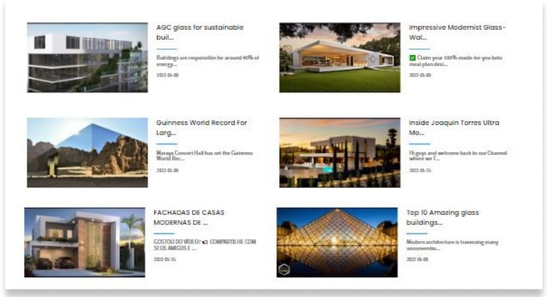
Gallery.js



Community.js



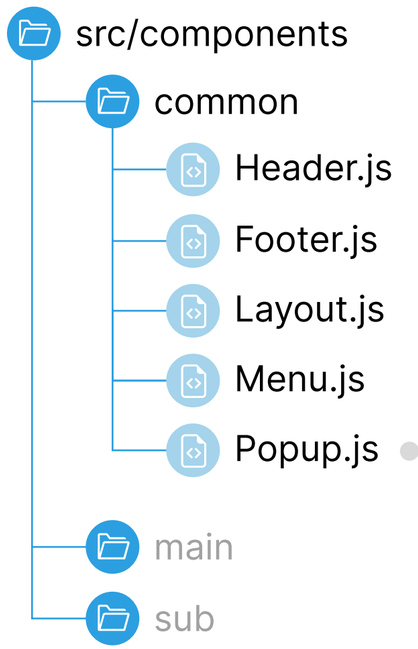
Youtube.js



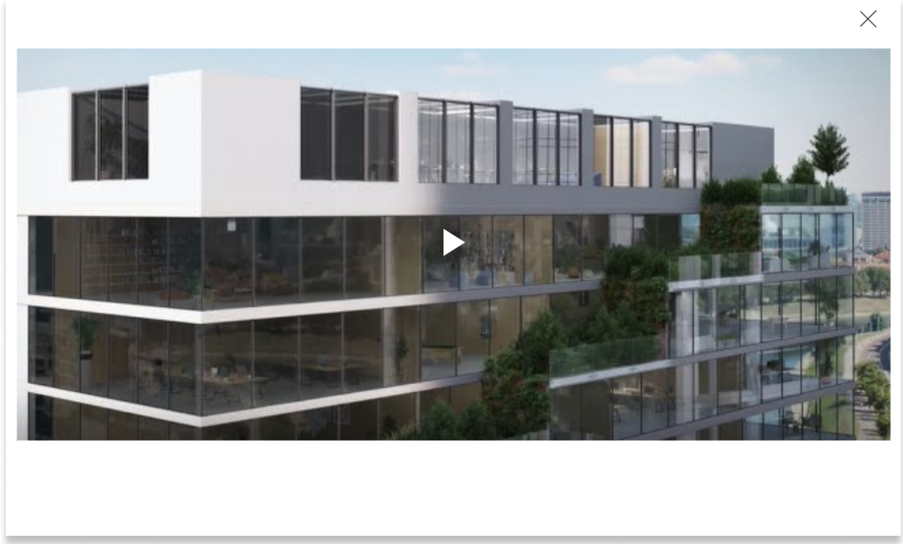
Department.js

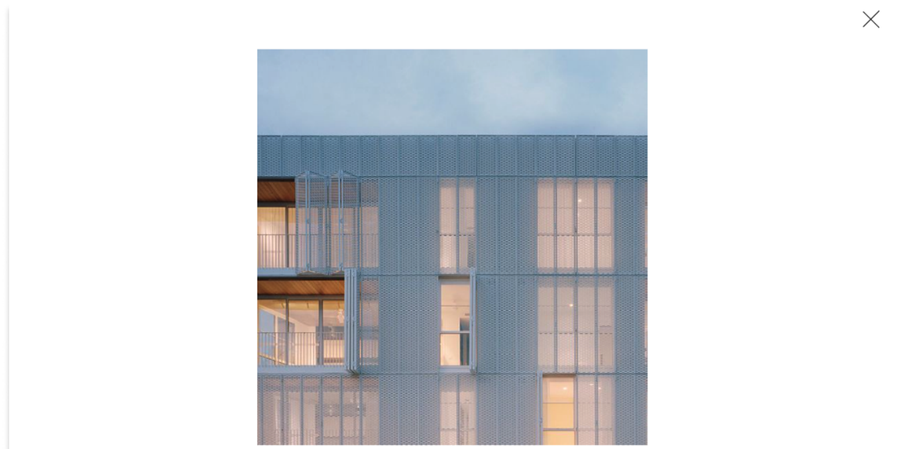
#1_5. COMMON STRUCTURE

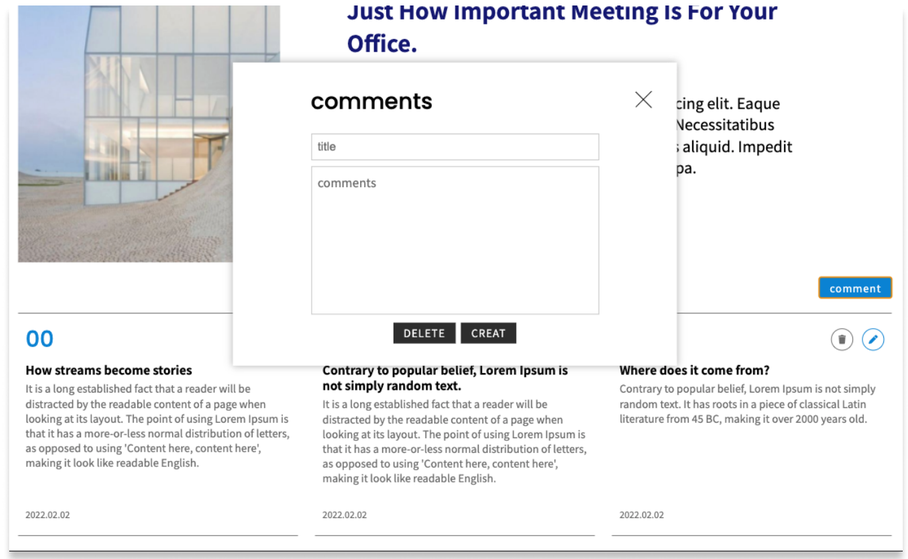
- **forwardRef**, **useImperativeHandle** Hook을 사용하여 하나의 레이아웃에 부모 컴포넌트에서 핸들링가능.
- props를 사용하용하여 페이지마다 다른 스타일 적용




```
const Popup = forwardRef((props, ref) => {
 const [open, setOpen] = useState(false);
 useImperativeHandle(ref, () => {
  return {
   open: () => setOpen(true),
   close: () => setOpen(false),
  };
 });
```







#1_6. FLICKR

- 이미지 공유 플러그인 FLICKR의 여러가지 메서드를 사용해 이미지 관리 및 검색 기능 구현

```
export const fetchGallery = async (opt) => {
 const key = '고유키 발급';
 const num = opt.count;
 const method_interest = 'flickr.interestingness.getList';
 const method_search = 'flickr.photos.search';
 const method_user = 'flickr.people.getPhotos';
 let url = '';

 if (opt.type === 'search') {
  url = `https://www.flickr.com/services/rest/?
method=${method_search}&per_page=${num}
&api_key=${key}
&nojsoncallback=1&format=json&tags=${opt.tag}`;
 }
 if (opt.type === 'user') {
  url = `https://www.flickr.com/services/rest/?
method=${method_user}&per_page=${num}
&api_key=${key}
&nojsoncallback=1&format=json&user_id=${opt.user}`;
 }

 return await axios.get(url);
};
```

#1_7. LOADING

- 큰 이미지와 비디오등 늦게 로드되는데 시간이 걸리는 이미지가 모두 로딩되기전에 로딩 처리
- 리액트가 로드되기 초기단계에서 작업해야하므로 index.html에 직접 코드 작성
- **new Promise()**인스턴스를 사용하여 해당 요소가 로드됐는지 판단후 **Promise.all()**객체를 사용해 모두 Promise가 순회 후 로딩이미지 삭제
- 로고를 svg로 제작하여 테두리에 animation을 효과를 줌

public

index.html


```
Promise.all([loadVid(), loadImg()]).then((result) => {
  if (result) {
    mask.classList.add('off');
    setTimeout(() => {
      mask.remove();
      defaults.remove();
    }, 2000);
  }
});

function createDOM() {
  vids.forEach(src => {
    tags += `<video src=${src}></video>`;
  })
  defaults.innerHTML = tags;
}

function loadVid() {
  return new Promise((res, rej) => {
    const vidDOM = defaults.querySelectorAll('video');
    let countVid = 0;

    vidDOM.forEach((vid) => {
      vid.onloadeddata = () => {
        countVid++;
        if (countVid === lenVid) {
          res(true);
        }
      }
    })
  })
}
```

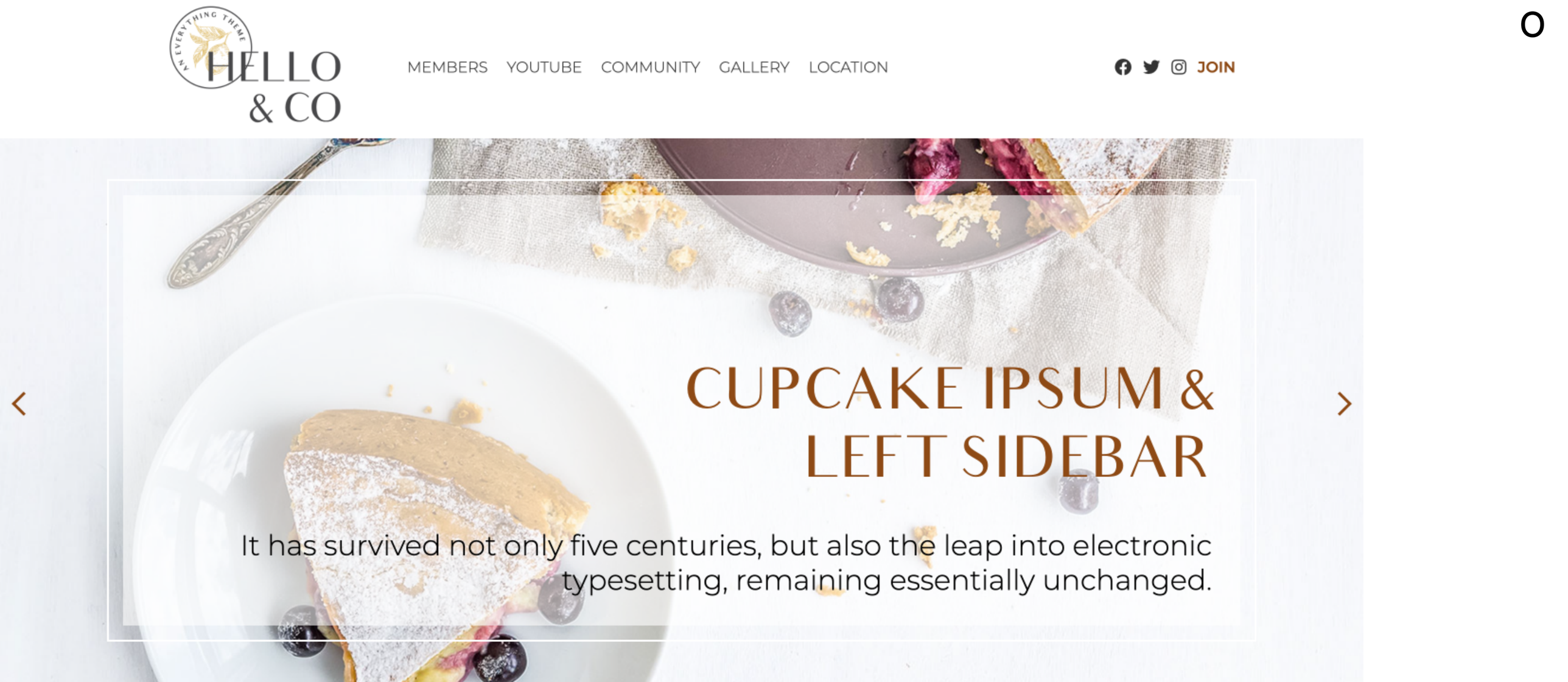
#2. INFO .

NAME : Hello & Co

URL : https://eurako.github.io/helloCo/

SPEC :

RESPONSIVE : PC, TABLET, MOBILE



#2_1. CONCEPT

- 기존 사이트의 메인을 커스터 마이징한 사이트로 따듯한 느낌을 살린 사이트

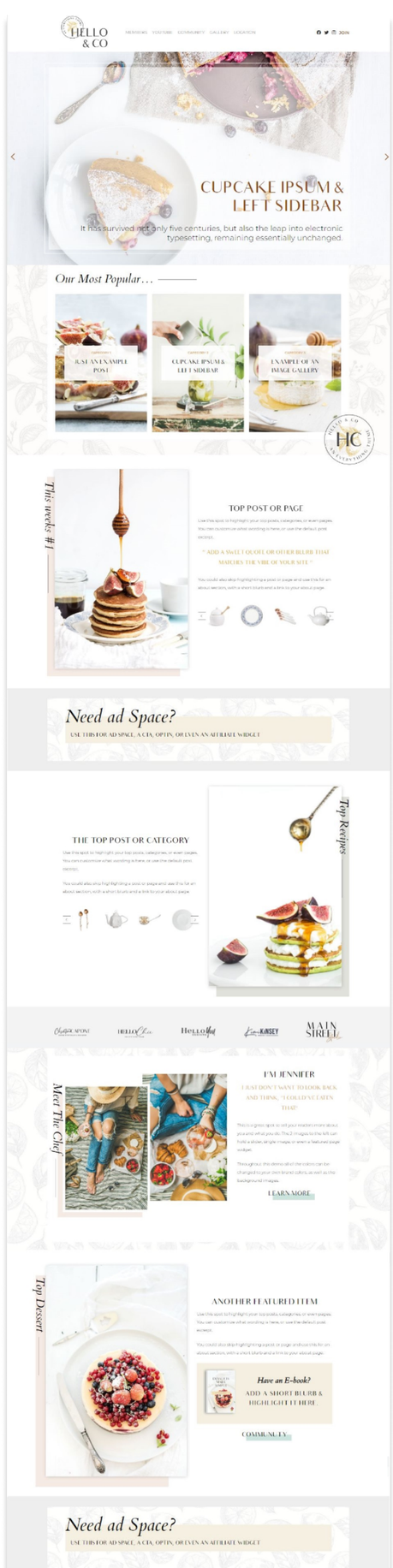
#FFFFFF #F6EBE4 #E0E1DA #DFB15B #8E4A15 #333333

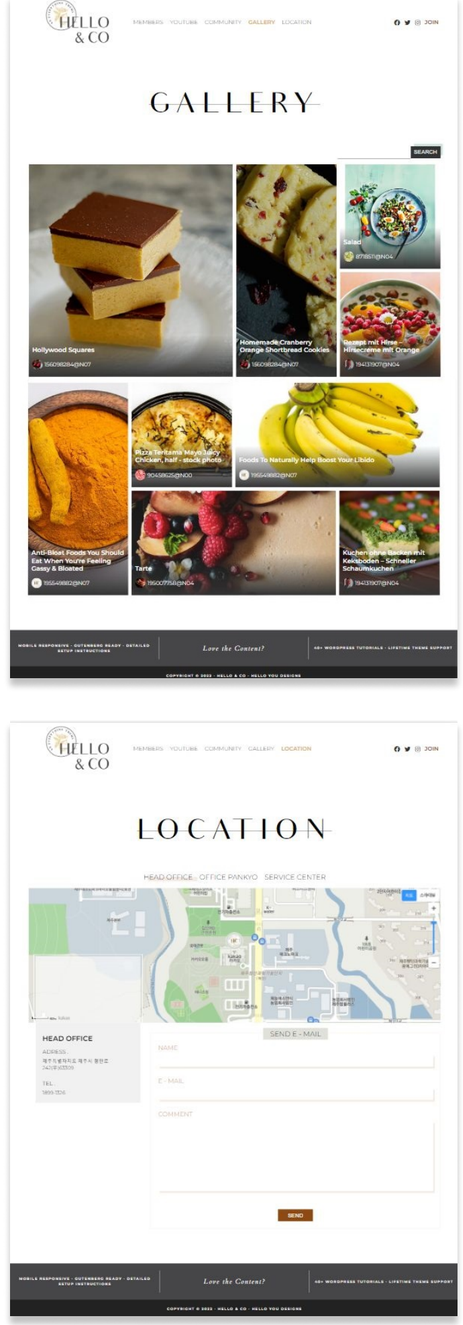
#2_2. STRUCTURE

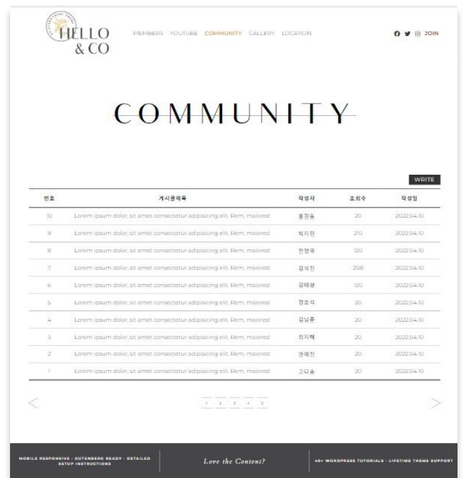
- scss와 jQuery없이 바닐라 스크립트와 ES6 문법만을 사용하여 프로젝트 구성

ROOT

- src
  - css — scss에서 최종 컴파일된 폴더
  - scss — css전처리기 scss파일들을 모아 놓은 폴더
  - js — script 및 json파일 관리
  - img — css전처리기 scss파일들을 모아 놓은 파일
  - fonts — 웹폰트로 지원되지 않은 폰트들을 따로 변환해 관리
- index.html
- communuty.html
- department.html
- qna.html
- gallery.html
- join.html
- location.html
- youtube.html







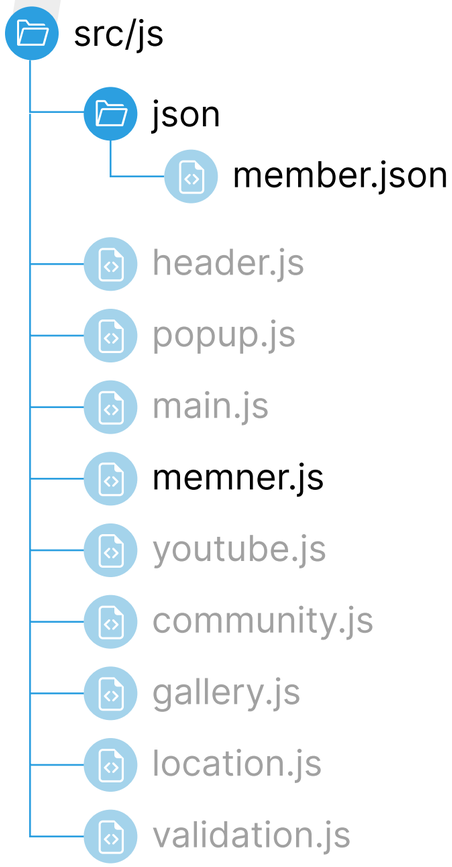
#2_3. JS STRUCTURE

- Vanilla JS와 ES6를 사용하여 개발하였으며 자주쓰는 모듈은 plugin화하여 재활용성을 높임


```
window.addEventListener('load', () => {
  new Banner('.banner1');
  new Banner('.banner2');
  new Scroll();
});

gnbBtn.addEventListener('click', e => {
  e.currentTarget.classList.add('on');
  gnbBox.classList.add('open');
  setTimeout(() => {
    gnbOverlay.classList.add('fade_bg');
    setTimeout(() => {
      gnbInner.classList.add('mob_open');
    }, 300)
  }, 100)

});

gnbCancle.addEventListener('click', e => {
  gnbInner.classList.remove('mob_open');
  setTimeout(() => {
    gnbOverlay.classList.remove('fade_bg');
    setTimeout(() => {
      gnbBtn.classList.remove('on');
      gnbBox.classList.remove('open');
    }, 300)
  }, 300)
});
```

#2_4. Fetch API

- 비동기 처리를 위해 callback의 단점을 보완한 Fetch API를 사용

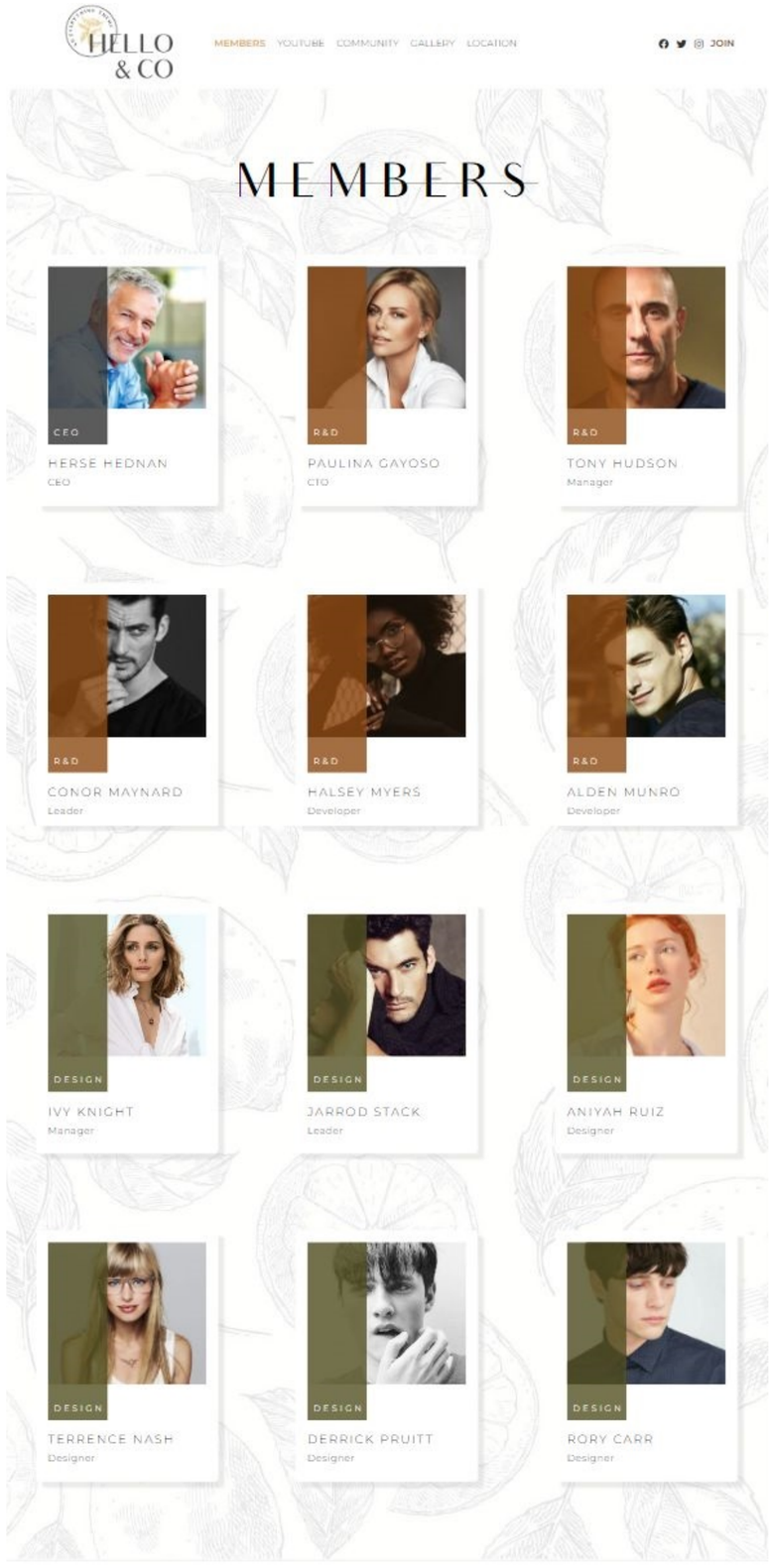


json/member.json

```
{
  "list": [{
      "name": "HERSE HEDNAN",
      "team": "ceo",
      "teamName": "CEO",
      "rank": "CEO",
      "img": "man6.jpg"
    },
        ⋮

  }]
}
```

member.js

```
fetch('src/js/json/member.json')
  .then((res) => {
    return res.json();
  })
  .catch((err) => {
    console.err('err')
  })
  .then((json) => {

    json.list.forEach((arr) => {
      makeMember(arr);
    });
  })
```



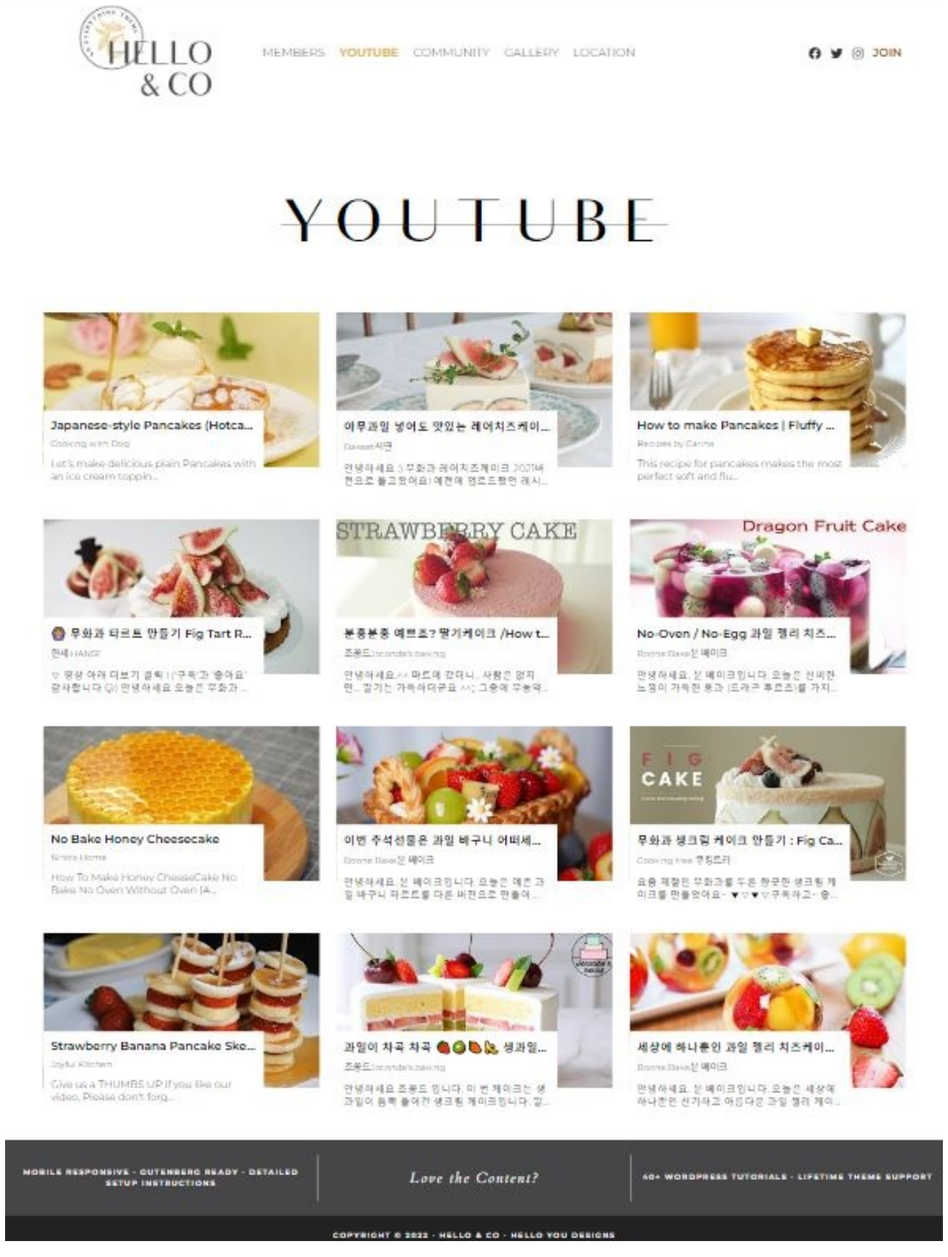
#2_5. YOUTUBE API

- youtube에 원하는 동영상만 모아 리스트로 만든 호출
- async await를 사용하여 데이터 비동기 처리

src/js

- header.js
- popup.js
- main.js
- memner.js
- youtube.js
- gallery.js
- location.js
- validation.js

```
const url = `불러올 URL`;

initList(url)

async function initList(url) {
  const data = await callData(url);
  creatList(data);
}

function callData(url) {
  return fetch(url)
    .then(data => {
      return data.json();
    })
    .catch(err => {
      console.error(err);
    })
    .then(json => {
      return json.items;
    })
}
```



#2_6. KAKAO MAP API

- 여러 지점을 클릭하면 지도의 위치 및 하단 정보 변경

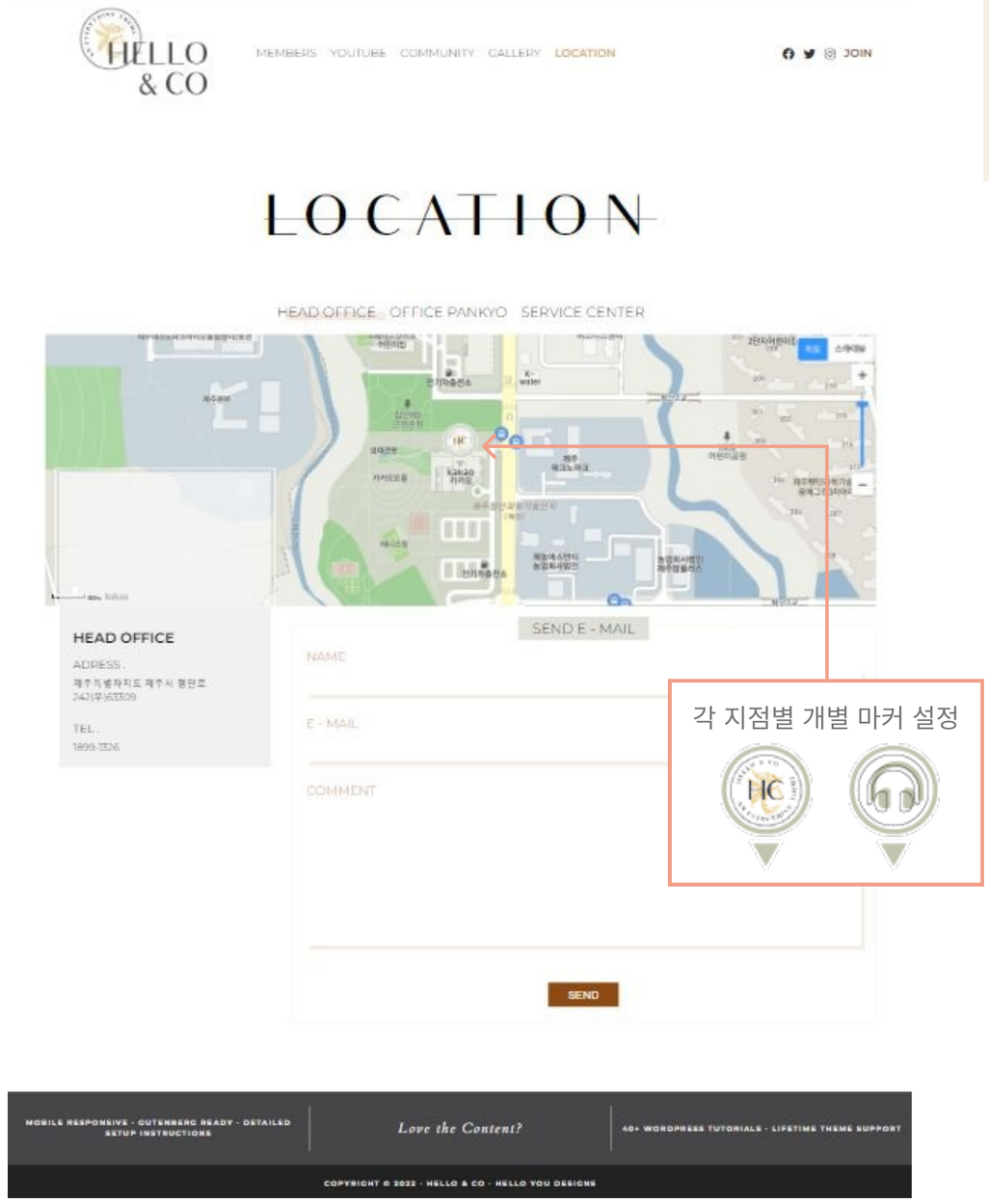
src/js
- header.js
- popup.js
- main.js
- memner.js
- youtube.js
- gallery.js
- location.js
- validation.js

<code>

```
let markerOption = [{
  title: 'HEAD OFFICE',
  latlng: new
kakao.maps.LatLng(33.450701,
126.570667),
  imgSrc: 'src/img/marker1.png',
  button: company_btns[0],
  address: '제주특별자치도 제주시 첨단
로 242(우)63309',
  tel: '1899-1326'
},
        ⋮
}]

 new kakao.maps.Marker({
      map: map,
      position: markerOption[i].latlng,
      title: markerOption[i].title,
      image: new
kakao.maps.MarkerImage(markerO
ption[i].imgSrc, markerSize,
markerPos)
    })
```

</code>

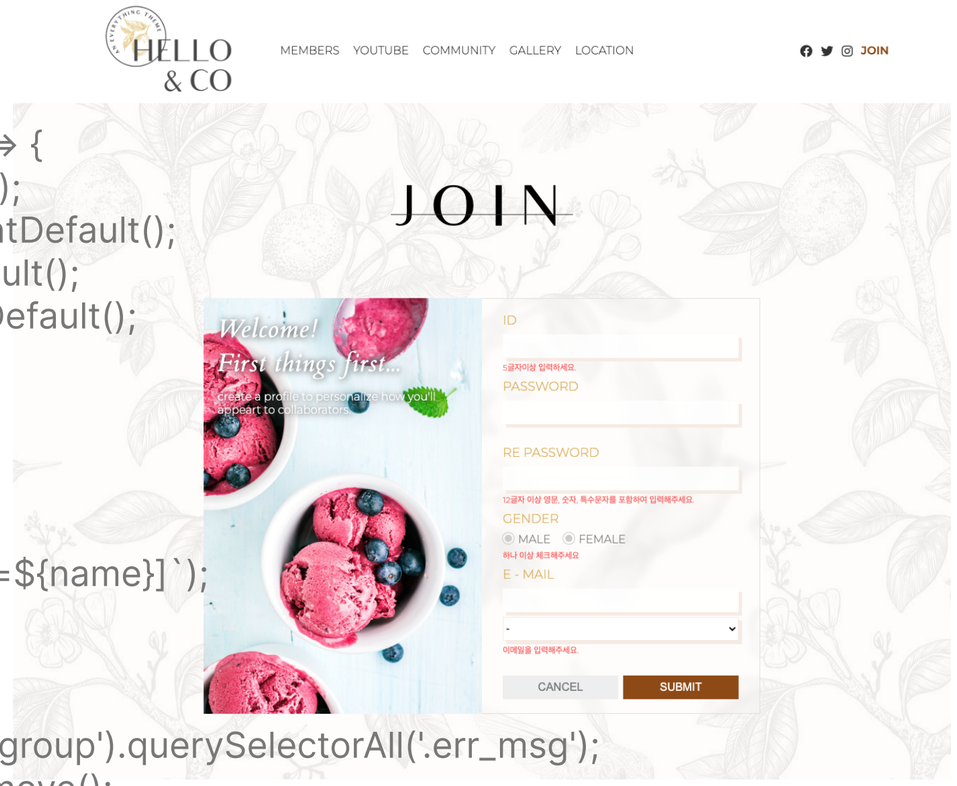
<script type="text/javascript" src="//dapi.kakao.com/v2/maps/sdk.js?appkey=<발급 key>"></script>



#2_7. VALIDATION

- 회원가입의 필수사항인 유효성 검증 구현

src/js
- header.js
- popup.js
- main.js
- memner.js
- youtube.js
- gallery.js
- location.js
- validation.js




```
btnSubmit.addEventListener('click', (e) => {
  if (!isTxt("userid", 5)) e.preventDefault();
  if (!isEmail('email', 'company')) e.preventDefault();
  if (!isChecked('gender')) e.preventDefault();
  if (!isPwd('pwd', 'pwd2', 12)) e.preventDefault();
});

function isTxt(name, len) {
  if (len === undefined) len = 5;

  let input = form.querySelector(`[name=${name}]`);
  let txt = input.value;

  if (txt.length >= len) {
    const errMsgs = input.closest('.input_group').querySelectorAll('.err_msg');
    if (errMsgs.length > 0) errMsgs[0].remove();
    return true;
  } else {
    const errMsgs = input.closest('.input_group').querySelectorAll('.err_msg');
    if (errMsgs.length > 0) errMsgs[0].remove();

    const errMsg = document.createElement('p');
    errMsg.classList.add('err_msg');
    errMsg.append(`${len}글자이상 입력하세요.`);
    input.closest('.input_group').append(errMsg);
    return false;
  }
}
```

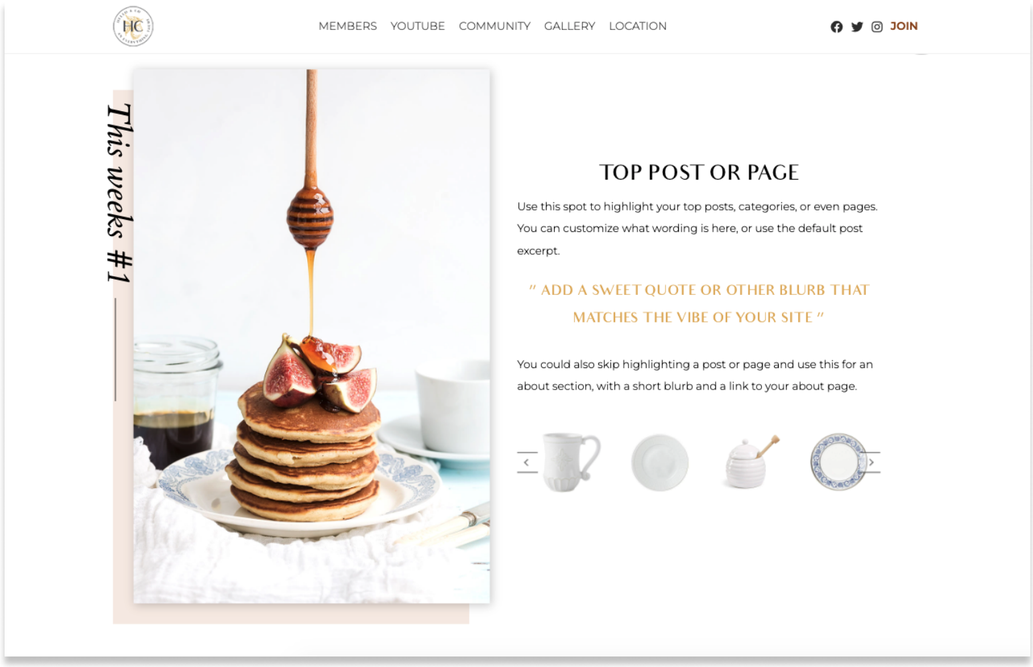
#2_8. SCSS STUCTURE

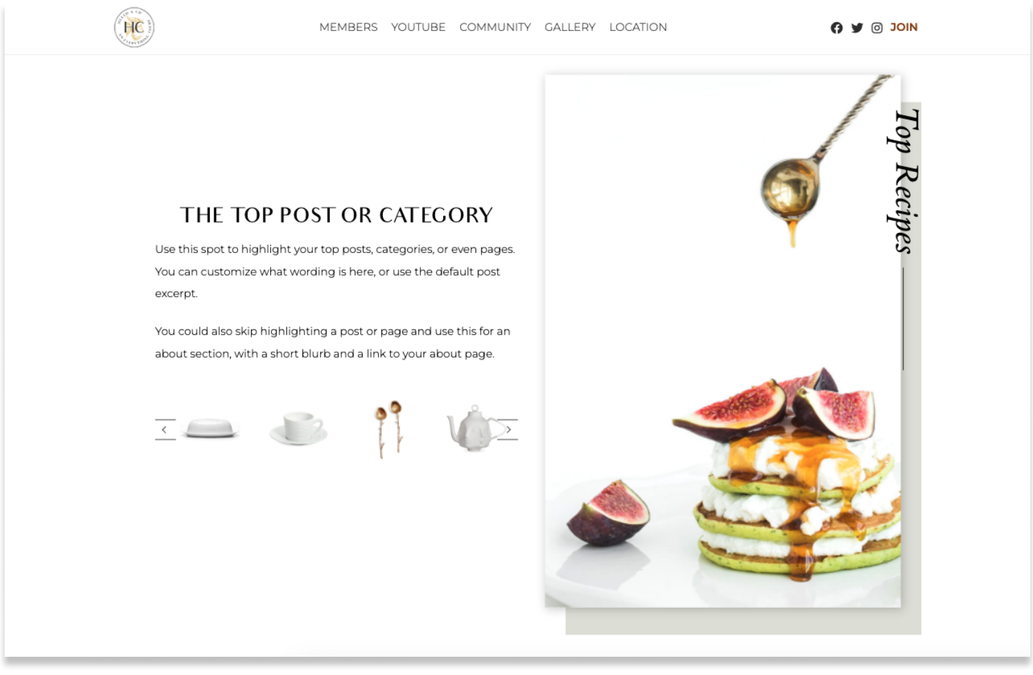
- css전처리기인 scss를 사용하여 코드의 가독성과 생산성을 높임

src/scss

- abstracts — 사이트에서 공통적으로 사용하는 디자인 폴더
  - _animation.scss — animation keyframe
  - _mixins.scss — 자주 사용하는 css 속성 변수
  - _variables.scss — 색상, 폰트등 변수
- base — 기본이 되는 폴더
  - _reset.scss — 브라우저에서 기본으로 적용되는 스타일 초기화
- components — 공통적으로 사용되는 스타일을 파일별로 분리
  - _input.scss
  - ⋮
- layout — 공통적으로 사용되는 레이아웃 폴더
  - _footer.scss
  - _header.scss
- pages — 페이지별 스타일 폴더
  - _main.scss
  - ⋮
- style.scss — 모든 스타일을 모아놓은 최종 scss파일

비슷한 레이아웃은 공통으로 처리 개발의 기간과 코드의 양을 줄여 생산성을 높임





#3. INFO .

NAME : CRUD 게시판

URL : https://react-jun.herokuapp.com/

SPEC :

RESPONSIVE : PC, TABLET

#3_1. CONCEPT

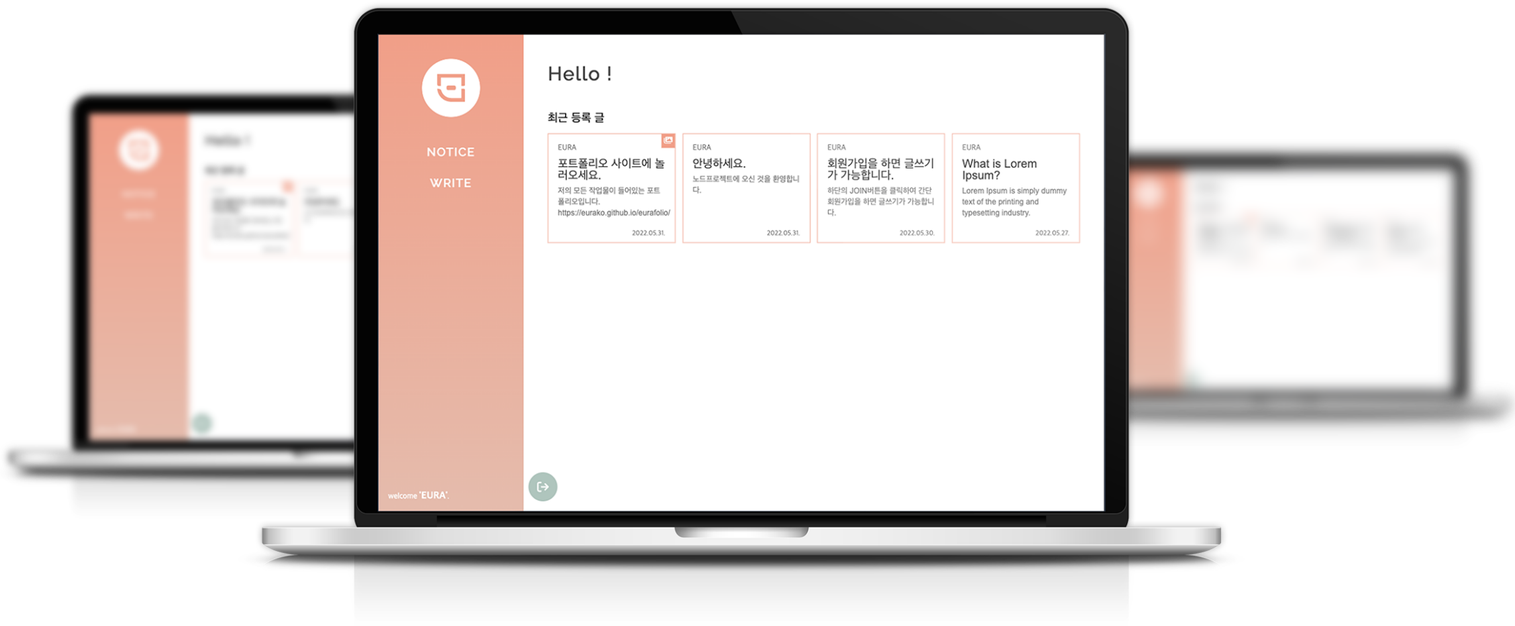
SERVER : node JS

DB : mongo DB

회원인증 : fire base

배포 : hefoku

이미지 외부 저장소 : naver cloud



#FFFFFF #E5BCAD #F19F88 #AFC5BD #4A7D7D #333333

#3_2. NODE STRUCTURE

- node 서버와 리액트 폴더를 따로 분리해서 관리

APP

img	node서버에 등록될 이미지 폴더
react	view 폴더
server	node js 폴더
cloud	heroku에서 서버가 꺼지면 등록된 이미지가 날아가는 현상이 생겨 네이버클라우드에서 이미지를 등록
config	노출되면 안되는 정보를 담는 배포와 개발 시 다른 값을 전달
model	모델 스키마 폴더
router	클라이언트의 요청을 전달할 router
index.js	최종 node의 명령을 수행할 파일
package.json	
Procfile	heroku 설정 파일

#3_3. JOIN

- 유효성 인증이 통과하면 firebase를 통해 인증완료 처리 후 redux를 통해 정보를 저장
- client에서 bodyparser로 전달받은 유저정보를 router에서 처리

react/compoments/Join.js

```
let createdUser = await firebase
 .auth()
 .createUserWithEmailAndPassword(email, pw1);

await createdUser.user.updateProfile({
 displayName: name,
});

const body = {
 email: createdUser.user.multiFactor.user.email,
 displayName: createdUser.user.multiFactor.user.displayName,
 uid: createdUser.user.multiFactor.user.uid,
};

 axios.post('/api/user/join', body).then((res) => {
  if (res.data.success) {
   navigate('/login');
  } else {
   return alert('회원가입에 실패했습니다.');
  }
 });
};
```

server/router/user.js

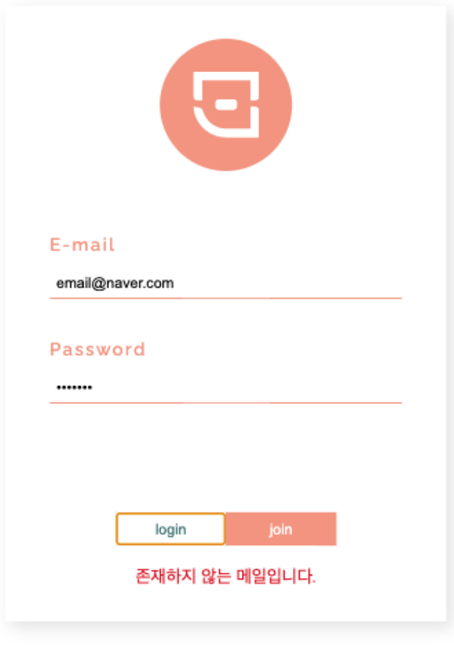
```
router.post('/join', (req, res) => {
 const temp = req.body;
 Counter.findOne({ name: 'counter' }).then((doc) => {
  temp.userNum = doc.userNum;

  const userData = new User(temp);
  userData
   .save()
   .then(() => {
    Counter.updateOne({ name: 'counter' }, { $inc: { userNum: 1 }
}).then(
     () => {
      res.status(200).json({ success: true });
     }
    );
   })
   .catch((err) => {
    console.log(err);
    res.status(400).json({ success: false });
   });
 });
});
```

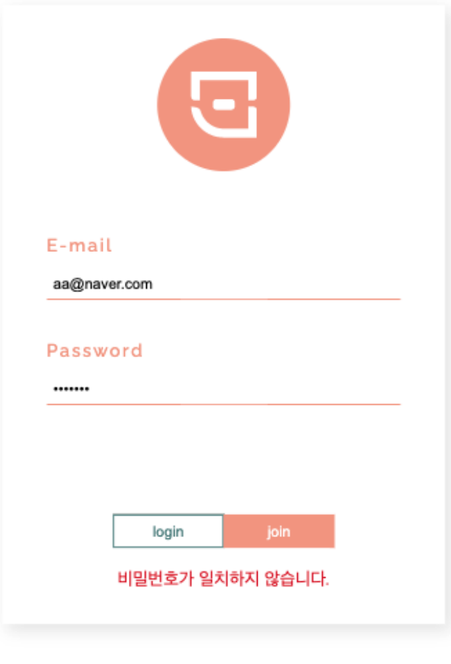
#3_4. LOGIN

- 로그인 인증이 되지않은 경우 에러별 메세지 출력

react/compoments/Login.js

```
const handleLogin = async () => {
 if (!(email && pw)) return alert('모든값을 입력하세요');

 try {
  await firebase.auth().signInWithEmailAndPassword(email, pw);
  navigate('/');
 } catch (err) {
  console.dir(err.code);
  if (err.code === 'auth/user-not-found') {
   setErrMsg('존재하지 않는 메일입니다.');
  } else if (err.code === 'auth/wrong-password') {
   setErrMsg('비밀번호가 일치하지 않습니다.');
  } else if (err.code === 'auth/invalid-email') {
   setErrMsg('이메일 형식이 아닙니다.');
  } else {
   setErrMsg('로그인에 실패했습니다.');
  }
 }
};
```



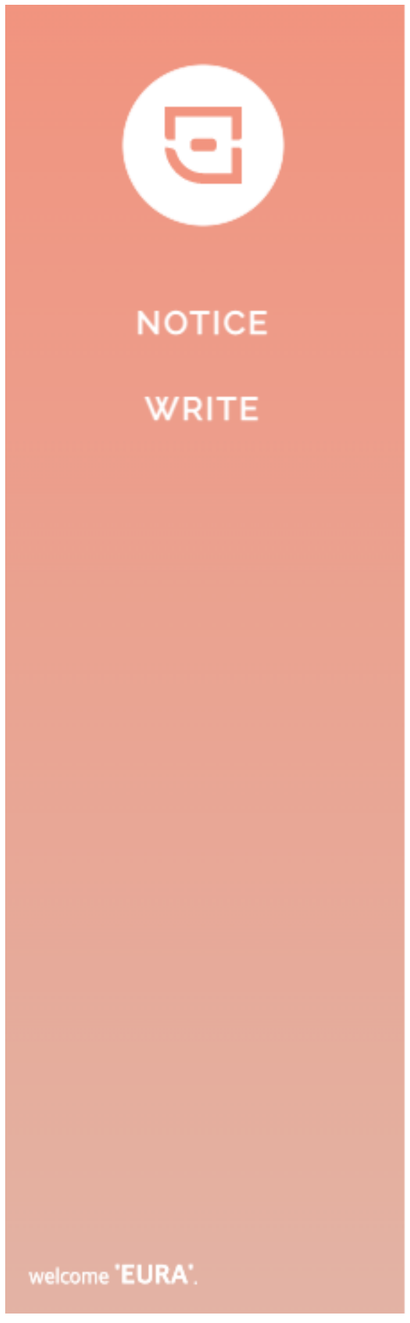


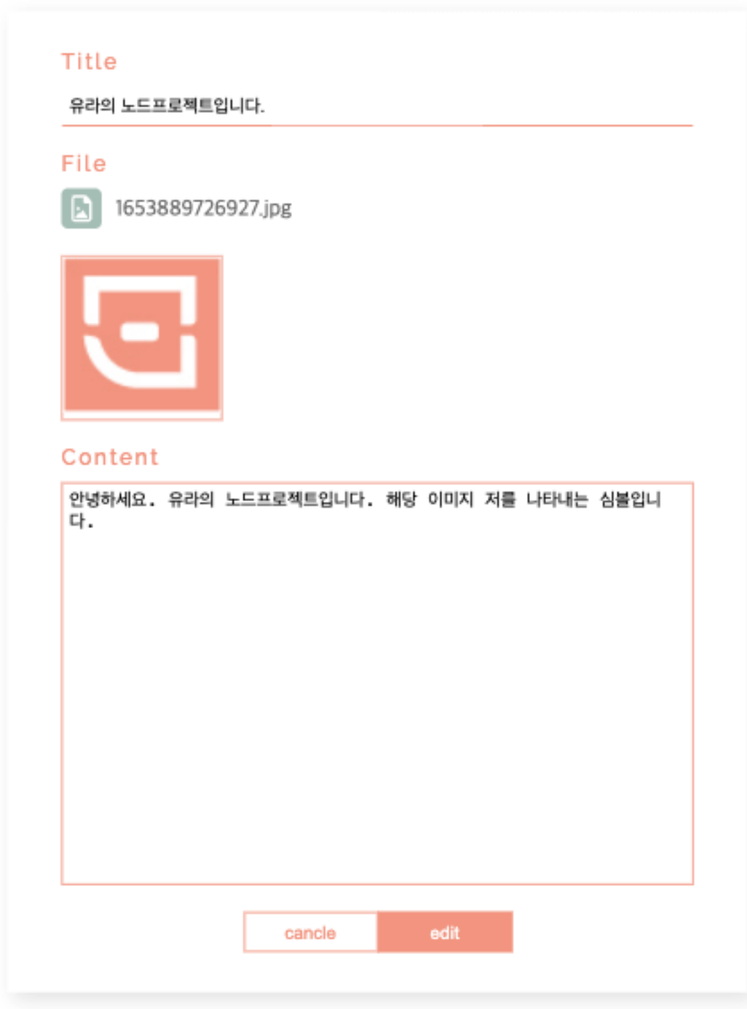
#3_5. WRITE & EDIT

- 컨텐츠 작성 및 수정
- haroku에서 서버가 닫히면 등록된 이미지가 삭제되는 이슈가 있어 naver cloud를 통해 이미지 업로드
- 이미지 등록시 미리보기 및 파일명 노출

react/compoments/Post.js

```
const onSubmit = () => {
 if (title === '' || content === '') {
  return alert('모든항목을 입력하세요');
 }
 const body = {
  title: title,
  content: content,
  img: img,
  imgName: imgName,
  uid: user.uid,
 };

 axios
  .post('/api/post/submit', body)
  .then((res) => {
   if (res.data.success) {
    alert('글작성이 완료되었습니다.');
    navigate('/list');
   } else {
    alert('글작성에 실패했어요');
   }
  })
  .catch((err) => {
   console.log(err);
  });
};
```





#3_6. LIST

- 작성된 글 리스트 화면에 출력
- 이미지가 있는 경우 아이콘 표시
- 검색 시 바디파서로 해당 내용 전달해 리스트 재출력

react/compoments/post/List.js

```
const getList = () => {
 let num = 0;
 if (props.count) num = props.count;
 const body = {
  sort: sort,
  search: search,
  count: num,
 };
 axios
  .post('/api/post/list', body)
  .then((res) => {
   // console.log(res);
   if (res.data.success) {
    // console.log(res.data);
    setList(res.data.postList);
    setLoaded(true);
   }
  })
  .catch((err) => {
   console.log(err);
  });
};
```

server/router/post.js

```
router.post('/list', (req, res) => {
 const sort = {};
 if (req.body.sort === 'new') {
  sort.createdAt = -1;
 }
 if (req.body.sort === 'old') {
  sort.createdAt = 1;
 }

 Post.find({
  $or: [
   { title: { $regex: req.body.search } },
   { content: { $regex: req.body.search } },
  ],
 })
  .populate('writer')
  .sort(sort)
  .limit(req.body.count)
  .exec()
  .then((doc) => {

   res.status(200).json({ success: true, postList: doc });
  })
  .catch((err) => {
   console.log(err);
   res.status(400).json({ success: false });
  });
});
```

#3_7. DETAIL

- 리스트 클릭 시 상세 페이지 노출
- 작성일과 최종수정일 따로 표시
- 작성자에게만 수정, 삭제버튼 노출

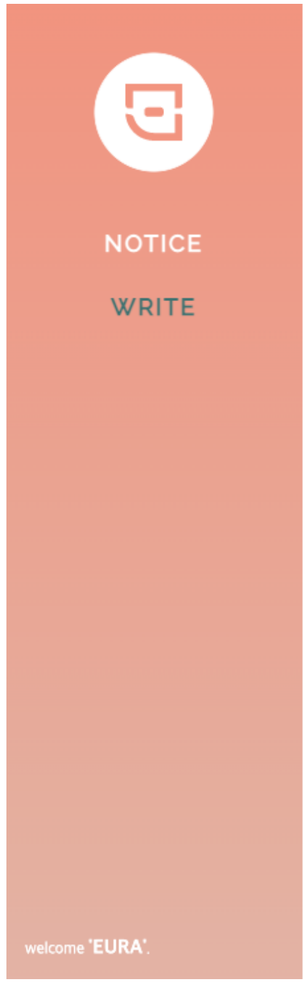
react/compoments/post/Detail.js

```
useEffect(() => {
 const body = {
  postNum: params.postNum,
 };

 axios
  .post('/api/post/detail', body)
  .then((res) => {
   if (res.data.success) {
    setDetail(res.data.post);
    console.log(res.data.post);
    setLoaded(true);
   }
  })
  .catch((err) => {
   console.log(err);
  });
}, []);
```

```
<p>{moment(detail.createdAt).format('L')}</p>

<p>{moment(detail.createdAt).format('a hh:mm:ss')}</p>
```

server/router/post.js

```
router.post('/detail', (req, res) => {
 Post.findOne({ postNum: Number(req.body.postNum) })
  .populate('writer')
  .exec()
  .then((doc) => {
   res.status(200).json({ success: true, post: doc });
  })
  .catch((err) => {
   console.log(err);
  });
});
```



View Detail