Contents

about

title: JEEP

url: https://fmdlivehj.github.io/jeep_portfolio

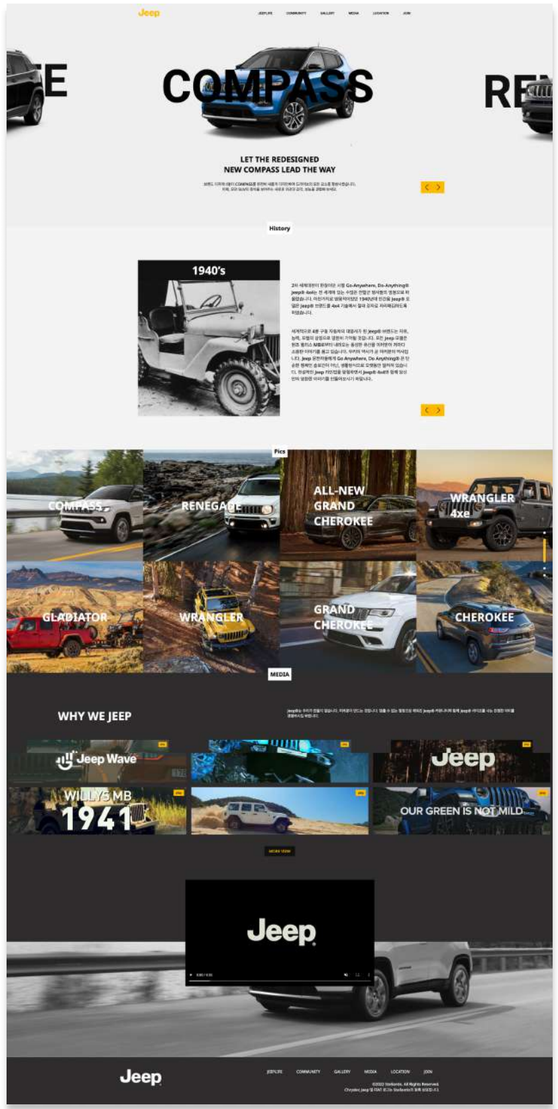
skill: react, redux, redux-saga, sass

layout

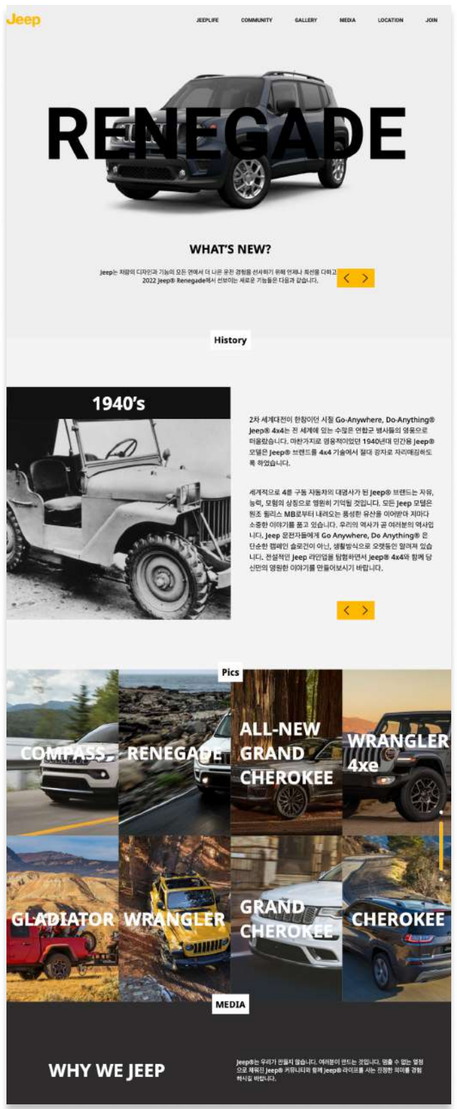
- 반응형 미디어 쿼리

반응형 구간 설정
$tablet : 1200px;
$mini : 900px;
#mobile : 539px

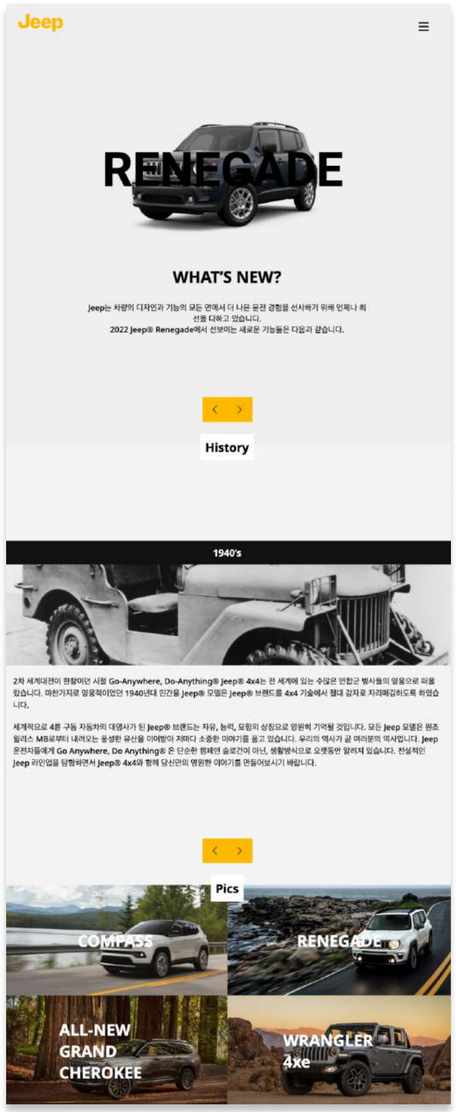
WEB



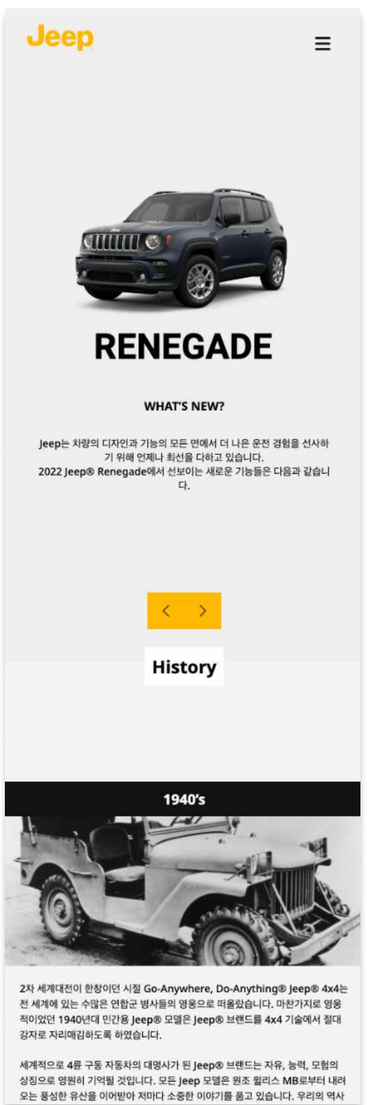
TABLET



MINI



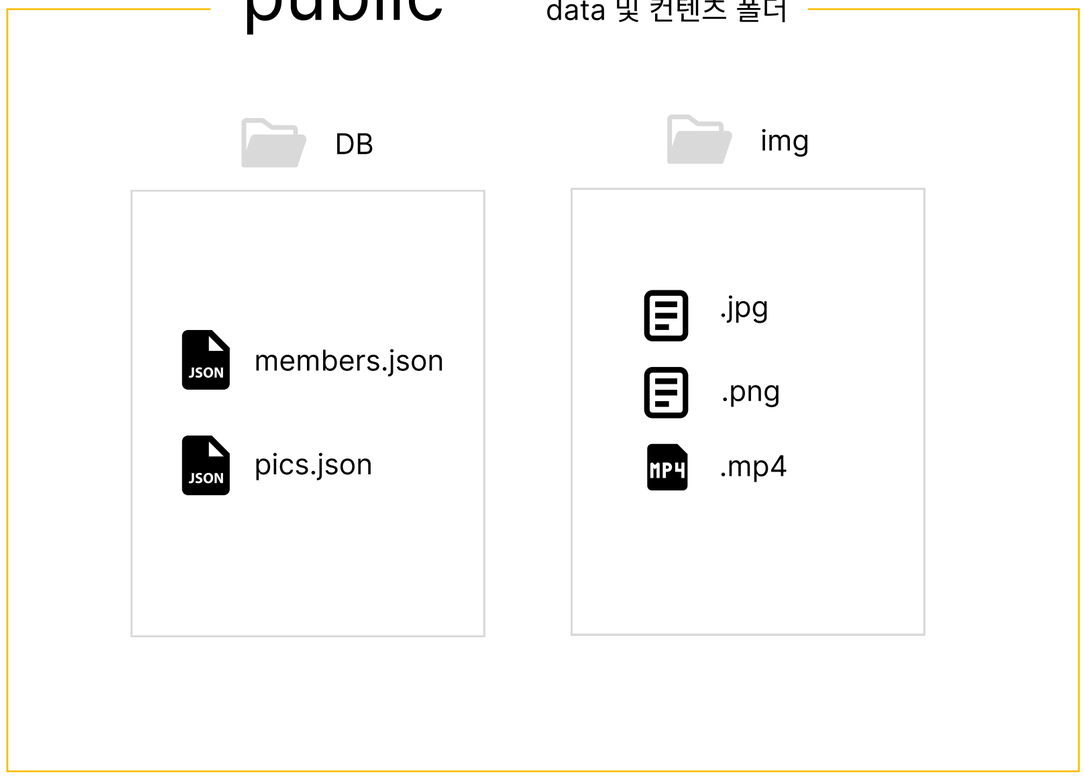
MOBILE

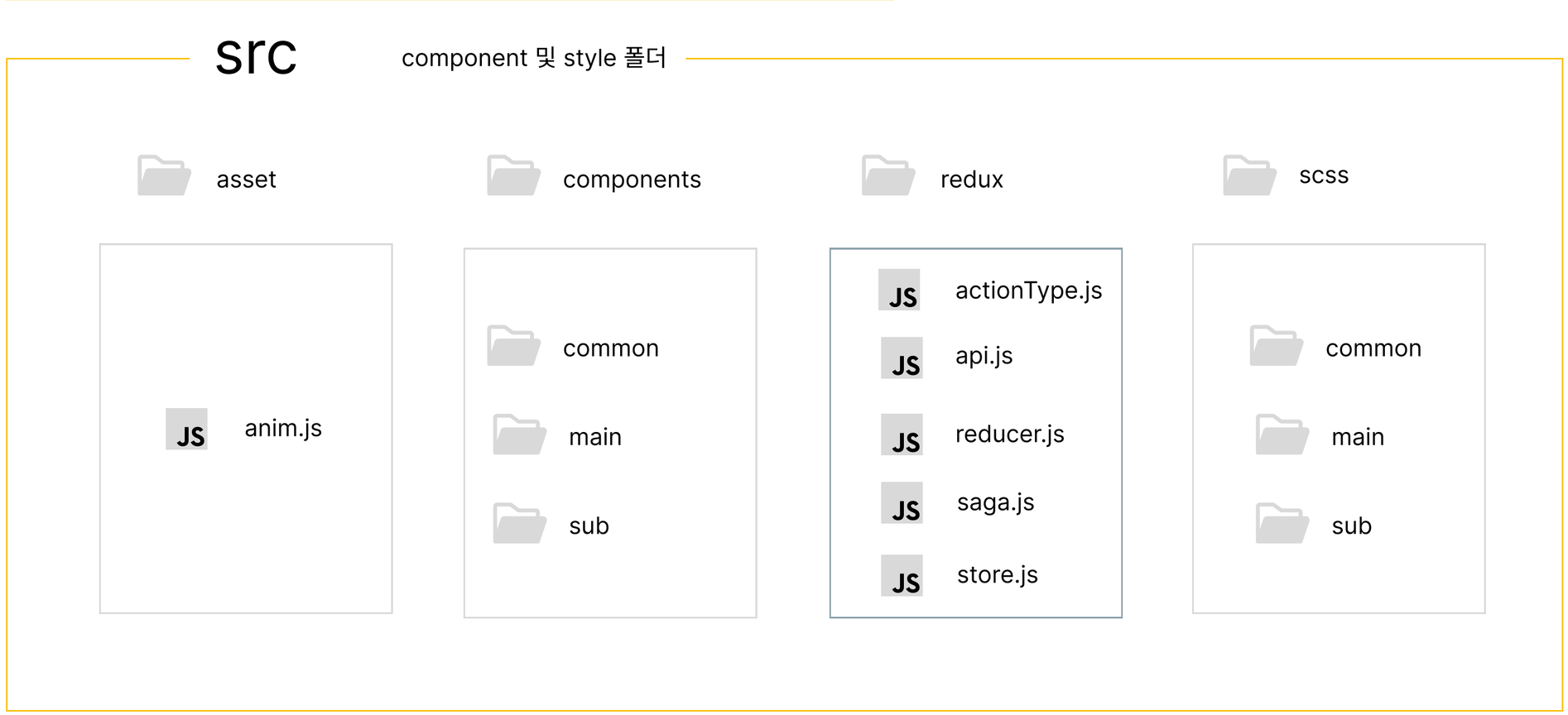


structure

- 컴포넌트 폴더 구조

- node_modules
- public
  - > DB
  - > img
    - index.html
- src
  - > asset
  - > components
  - > redux
  - > scss
    - App.js
    - index.js
- package.json



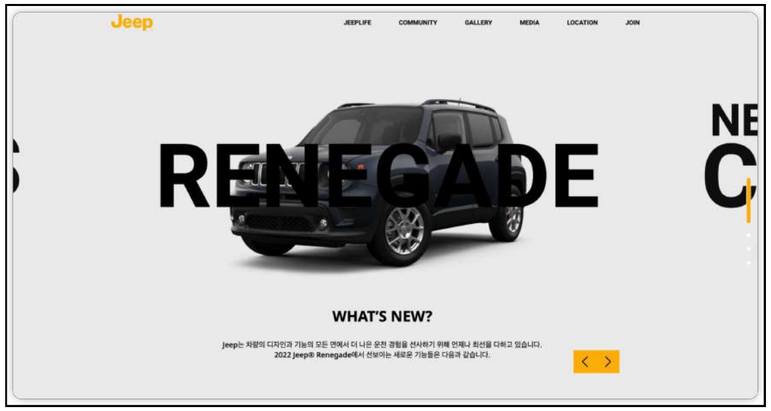


main

components

main

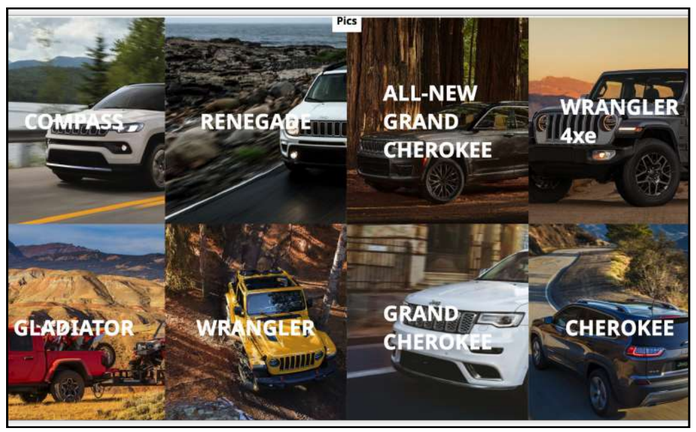
- Btns.js
- History.js
- Main.js
- Pics.js
- Vids.js
- Visual.js



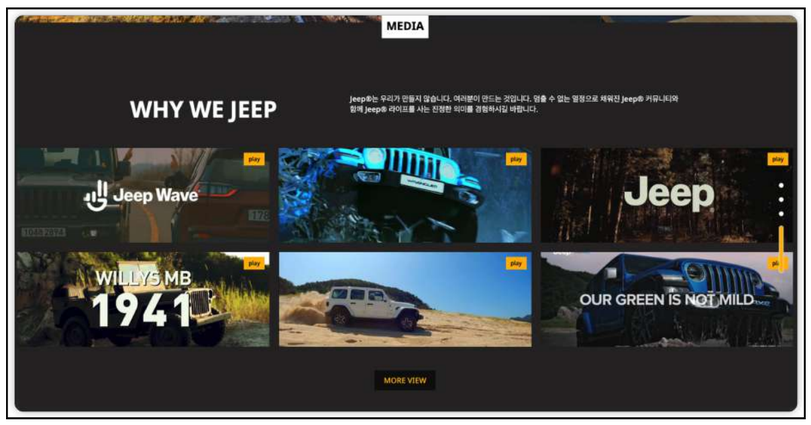
컨텐츠별 텍스트 svg 사용
swiper slider 사용

forwardRef사용한 pop업 생성

react-saga 사용,
members.json 데이터 불러옴



react-saga 사용,
pics.json 데이터 불러옴
컨텐츠 hover시 영상재생효과



react-saga 사용,
youtube재생목록 데이터 불러옴
클릭시 팝업생성

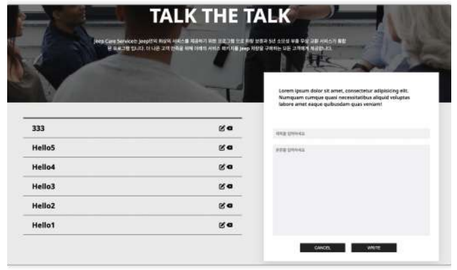
sub

components

sub

- Community.js
- Gallery.js
- Jeeplife.js
- Join.js
- Location.js
- Media.js
- MediaSlider.js

localstoage 사용



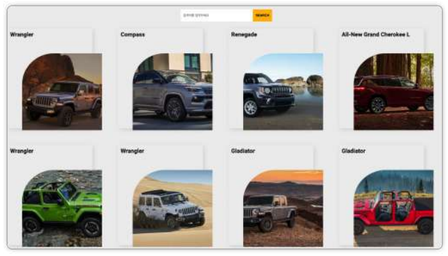
&. localstorage에서 데이터를 추가 및 불러옴

```
const getLocalData = () => {
 const data = localStorage.getItem('post');
 const dummyPosts = [
  { title: 'Hello5', content: 'Here comes description in detail.' },
....
 const [Posts, setPosts] = useState(getLocalData());

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

&. members.json 데이터 불러옴 redux saga 이용해서 api.js로 연결해줌

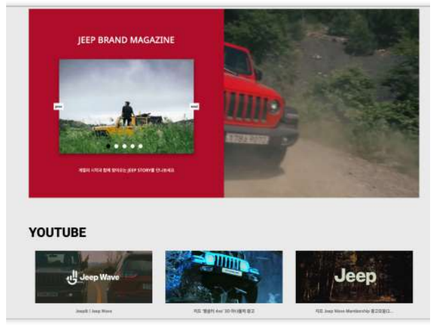
```
export const fetchYoutube = async () => {
 const key = 'AIzaSyC77Pd__ju0Wqx_Umc-IuW7Cn2mWi_HVsk';
 const playlist = 'PLooTX0MOEe6OAucsxOqZTV72g8pieLz9z';
 const num = 9;
 const url = `https://www.googleapis.com/youtube/v3/playlistItems?
part=snippet&key=${key}&playlistId=${playlist}&maxResults=${num}`;

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

react-saga 사용



&. flickr.com 와 연동하여 데이터 불러옴 redux saga 이용해서 api.js로 연결해줌

```
export const fetchMember = async () => {
 const url = `${process.env.PUBLIC_URL}/DB/members.json`;

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

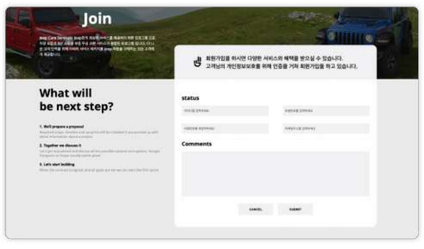


MediaSlider에 slider 연결

&. youtube 와 연동하여 데이터 불러옴 redux saga 이용해서 api.js로 연결해줌

```
export const fetchYoutube = async () => {
 const key = 'AIzaSyC77Pd__ju0Wqx_Umc-IuW7Cn2mWi_HVsk';
 const playlist = 'PLooTX0MOEe6OAucsxOqZTV72g8pieLz9z';
 const num = 9;
 const url = `https://www.googleapis.com/youtube/v3/playlistItems?
part=snippet&key=${key}&playlistId=${playlist}&maxResults=${num}`;

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

form data 인증처리



kakao api사용

common

- 메인과 서브 공통 컴포넌트

components

common

- Footer.js
- Header.js
- Layout.js
- Menu.js
- Popup.js

App.js

App

메인,서브 분리

```
<>
 <Switch>
  <Route exact path='/'>
   <Main />
  </Route>

  {/* 서브용 header */}
  <Route path='/' render={() => <Header type={'sub'} />} />
 </Switch>

 <Route path='/jeeplife' component={Jeeplife} />
 <Route path='/community' component={Community} />
 <Route path='/gallery' component={Gallery} />
 <Route path='/media' component={Media} />
 <Route path='/location' component={Location} />
 <Route path='/join' component={Join} />

 <Footer />
</>
```

layout

서브페이지 공통 모듈화

```
function Layout({ name, children }) {
 const frame = useRef(null);

 useEffect(() => {
  frame.current.classList.add('on');
 }, []);

 return (
  <section className={`content ${name}`} ref={frame}>
   <div className='container'>
    {children}
   </div>
  </section>
 );
}
```

menu

AnimatePresence, motion 사용

```
<AnimatePresence>
 {Open && (
  <motion.nav
   id='mobileGnb'
   initial={{ x: -320, opacity: 0 }}
   animate={{
    x: 0,
    opacity: 1,
    transition: { duration: 0.5 },
   }}
   exit={{ x: -320, opacity: 0 }}
   onClick={() => setOpen(!Open)}>
   <h1>
    <NavLink exact to='/'>
     JEEP
    </NavLink>
   </h1>
```

Popup

forwardRef,
useImperativeHandle
사용 외부로 반환

```
<AnimatePresence>
 {Open && (
  <motion.nav
   id='mobileGnb'
   initial={{ x: -320, opacity: 0 }}
   animate={{
    x: 0,
    opacity: 1,
    transition: { duration: 0.5 },
   }}
   exit={{ x: -320, opacity: 0 }}
   onClick={() => setOpen(!Open)}>
   <h1>
    <NavLink exact to='/'>
     JEEP
    </NavLink>
   </h1>
```

footer

main , sub 공통

```
function Footer() {
 return (
  <footer>
   <div className='inner'>
    <h1>logo</h1>
    <div>
     <ul>
      <li>
       <NavLink to='/Jeeplife'>JEEPLIFE</NavLink>
      </li>
```

header

main , sub 공통

```
function Header({ type }) {
 const menu = useRef(null);
 const style = { color: '#ffba00' };

 return (
  <header className={type}>
   <div className='inner'>
    <h1>
     <Link to='/'>JEEP</Link>
    </h1>
    <ul id='gnb'>
     <li>
      <NavLink activeStyle={style} to='/Jeeplife'>
       JEEPLIFE
      </NavLink>
     </li>
```

index

- 리액트 실행 페이지

동적으로 배열에 있는 이미지, 영상 소스를 가지고 index.html에 DOM강제생성

DOM이 생성될때 영상, 소스이미지가 브라우저에 캐싱되는 동안에는 loading화면 보여줌

배열에 있는 모든 데이터가 캐싱완료되면 마스크와 강제로 만들어진 DOM프레임인 defaults를 모두 제거

public

index.html

캐싱 img, mp4

```
const baseURL = 'https://fmdlivehj.github.io/jeep_portfolio'
  const imgs = [
    baseURL + '/img/media/story02.jpg',
    baseURL + '/img/media/story01.jpg',
];


 const vids = [
   baseURL + '/img/jeeplife/members_video.mp4',
   baseURL + '/img/media/story.mp4',

];
```

DOM생성함수

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

동기화처리

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

로딩되면 프로미스 객체로 true반환 함수

```
function loadImg() {
  return new Promise((res, rej) => {
    let countImg = 0;
    const imgDOM = defaults.querySelectorAll('img');

    imgDOM.forEach((img) => {
      img.onload = () => {
        countImg++;

        if (countImg === lenImg) {
          res(true);
        }
      }
    })
  })
}
```