



실증 아카데미 리액트 특강

# fetch API를 통해 외부 API 호출하기



일시 2021년 01월 19일

주최 동아대학교





# CONTENTS—

## 01 JavaScript 비동기 통신

AJAX  
XMLHttpRequest  
fetch API

## 02 외부 API 요청

네이버 개발자 센터 등록  
API 키 발급  
fetch API 코드 작성

## 03 API 요청 시 로딩

useState를 통한 로딩

## 04 외부 데이터 렌더링

요청 받은 데이터 state로 관리



01

# JavaScript 비동기 통신

AJAX

XMLHttpRequest

fetch API

# AJAX

## Asynchronous JavaScript and XML

- JavaScript와 XML을 이용한 비동기적 정보 교환 기법
  - 기술이름은 XML이지만, Text 등 다른 데이터도 가능하며, 요즘은 JSON 사용
  - AJAX를 사용하지 않으면 브라우저가 페이지를 깜빡거리며(새로고침) 처음부터 다시 렌더링
  - 서버에서는 HTML 문서가 아닌, 순수한 데이터만을 응답해줄 수 있음
  - 동적 페이지를 만들기 때문에 검색 엔진에 내용 검색이 안 된다는 단점이 있었음(SEO)
  - 비동기 데이터 전송 기술이지만 양방향 기술이 아니라, 요청 - 응답 사이클이 지나면 소켓을 닫아버린다는 단점 => 웹 소켓 기술로 보완
- AJAX로 구현한 Long polling(socket.io에서 지원) 이라는 기술도 있음

JavaScript에서는 XMLHttpRequest, fetch를 지원



# XMLHttpRequest(XHR)

## 과거 JavaScript에서 AJAX를 지원하기 위한 기술

- MDN 문서: <https://developer.mozilla.org/ko/docs/Web/API/XMLHttpRequest>
- 과거의 방식이라 코드가 난잡하여 가독성이 나쁨
- XML과 Text를 지원하기 때문에 요청과 응답 모두 JSON 파일인 경우 JSON.parse 함수 사용
- 에러 발생을 핸들링해주지 않아서 직접 작성해야 함
- Promise가 아닌 비동기 콜백 방식이라 최신 JavaScript와의 궁합이 좋지 않음
- XMLHttpRequest 객체를 생성하여 객체의 정보를 주입한 후 통신 요청

API 요청 사이트: <https://reqres.in/>



## - /src/components/RequestText.jsx 파일 생성 후 코드 작성

```
import { useCallback, useEffect } from "react";
const URL = "https://reqres.in/api/users";
const RequestTest = () => {
  const callApiGetXhr = useCallback(() => {
    const xhr = new XMLHttpRequest();
    xhr.open("GET", URL, true);
    xhr.onreadystatechange = () => {
      if (xhr.readyState === XMLHttpRequest.DONE) {
        const { status } = xhr;
        console.log(xhr);
        if (status === 0 || (status >= 200 && status < 400)) {
          console.log(JSON.parse(xhr.responseText));
        }
      }
    };
    xhr.send();
  }, []);
  useEffect(() => {
    callApiGetXhr();
  }, [callApiGetXhr]);
  return <></>;
};
export default RequestTest;
```

## - App.js에서 import 후 렌더

```
import { StyleApp, StyleAppContent } from "./App.style";
import RequestTest from "./components/RequestTest";

const App = () => {
  return (
    <StyleApp>
      <StyleAppContent>
        <RequestTest />
      </StyleAppContent>
    </StyleApp>
  );
};
export default App;
```

API 요청 사이트: <https://reqres.in/>

# 응답 결과 확인

RequestTest.jsx:13

```
XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, onreadystatechange: f, ...} ⓘ
  onabort: null
  onerror: null
  onload: null
  onloadend: null
  onloadstart: null
  onprogress: null
  ▶ onreadystatechange: () => {...}
  ontimeout: null
  readyState: 4
  response: "{\"page\":1,\"per_page\":6,\"total\":12,\"total_pages\":2,\"data\": [{\"id\":1,\"email\":...
  responseText: "{\"page\":1,\"per_page\":6,\"total\":12,\"total_pages\":2,\"data\": [{\"id\":1,\"emai...
  responseType: ""
  responseURL: "https://reqres.in/api/users"
  responseXML: null
  status: 200
  statusText: ""
  timeout: 0
  ▶ upload: XMLHttpRequestUpload {onloadstart: null, onprogress: null, onabort: null, onerror: null, on...
    withCredentials: false
  ▶ [[Prototype]]: XMLHttpRequest
```

RequestTest.jsx:15

```
{page: 1, per_page: 6, total: 12, total_pages: 2, data: Array(6), ...} ⓘ
  ▶ data: (6) [{...}, {...}, {...}, {...}, {...}, {...}]
  page: 1
  per_page: 6
  ▶ support: {url: 'https://reqres.in/#support-heading', text: 'To keep ReqRes free, contributions towa...
    total: 12
    total_pages: 2
  ▶ [[Prototype]]: Object
```

API 요청 사이트: <https://reqres.in/>

# fetch

## 브라우저에서 정식 지원하는 HTTP 요청 인터페이스

- MDN 문서: [https://developer.mozilla.org/ko/docs/Web/API/Fetch\\_API](https://developer.mozilla.org/ko/docs/Web/API/Fetch_API)
- XMLHttpRequest보다 강력하고 유연한 조작 가능
- XML과 Text, JSON 모두 지원.
- Promise에서 에러가 구현되어 try catch로 에러 핸들링 가능
- Promise로 구현되었기 때문에 최신 JavaScript와 궁합이 좋음
- 객체를 따로 생성할 필요 없이 바로 요청 가능

※ 구형 브라우저에서는 제대로 지원되지 않는 경우가 간혹 있음(IE)





## - /src/components/RequestText.jsx 파일 생성 후 코드 작성

```
import { useCallback, useEffect } from "react";
const URL = "https://reqres.in/api/users";
const RequestTest = () => {
  const callApiFetch = useCallback(() => {
    fetch(URL)
      .then((response) => {
        response.json().then((data) => console.log(data));
      })
      .catch((error) => {
        console.log(error);
      });
  }, []);
  useEffect(() => {
    callApiFetch();
  }, [callApiFetch]);
  return <></>;
};
export default RequestTest;
```

```
▼ {page: 1, per_page: 6, total: 12, total_pages: 2, data: Array(6), ...} ⓘ
  ▼ data: Array(6)
    ▶ 0: {id: 1, email: 'george.bluth@reqres.in', first_name: 'George', last_name: 'Bluth Jr'}
    ▶ 1: {id: 2, email: 'janet.weaver@reqres.in', first_name: 'Janet', last_name: 'Weaver'}
    ▶ 2: {id: 3, email: 'emma.wong@reqres.in', first_name: 'Emma', last_name: 'Wong'}
    ▶ 3: {id: 4, email: 'eve.holt@reqres.in', first_name: 'Eve', last_name: 'Holt', avatar: 'https://reqres.in/img/avatar/4.jpg'}
    ▶ 4: {id: 5, email: 'charles.morris@reqres.in', first_name: 'Charles', last_name: 'Morris', avatar: 'https://reqres.in/img/avatar/5.jpg'}
    ▶ 5: {id: 6, email: 'tracey.ramos@reqres.in', first_name: 'Tracey', last_name: 'Ramos', avatar: 'https://reqres.in/img/avatar/6.jpg'}
    length: 6
    ▶ [[Prototype]]: Array(0)
  page: 1
  per_page: 6
  ▼ support:
    text: "To keep ReqRes free, contributions towards server costs are appreciated!"
    url: "https://reqres.in/#support-heading"
    ▶ [[Prototype]]: Object
  total: 12
  total_pages: 2
  ▶ [[Prototype]]: Object
```

API 요청 사이트: <https://reqres.in/>

# 02

## 외부 API 요청

네이버 개발자 센터 등록

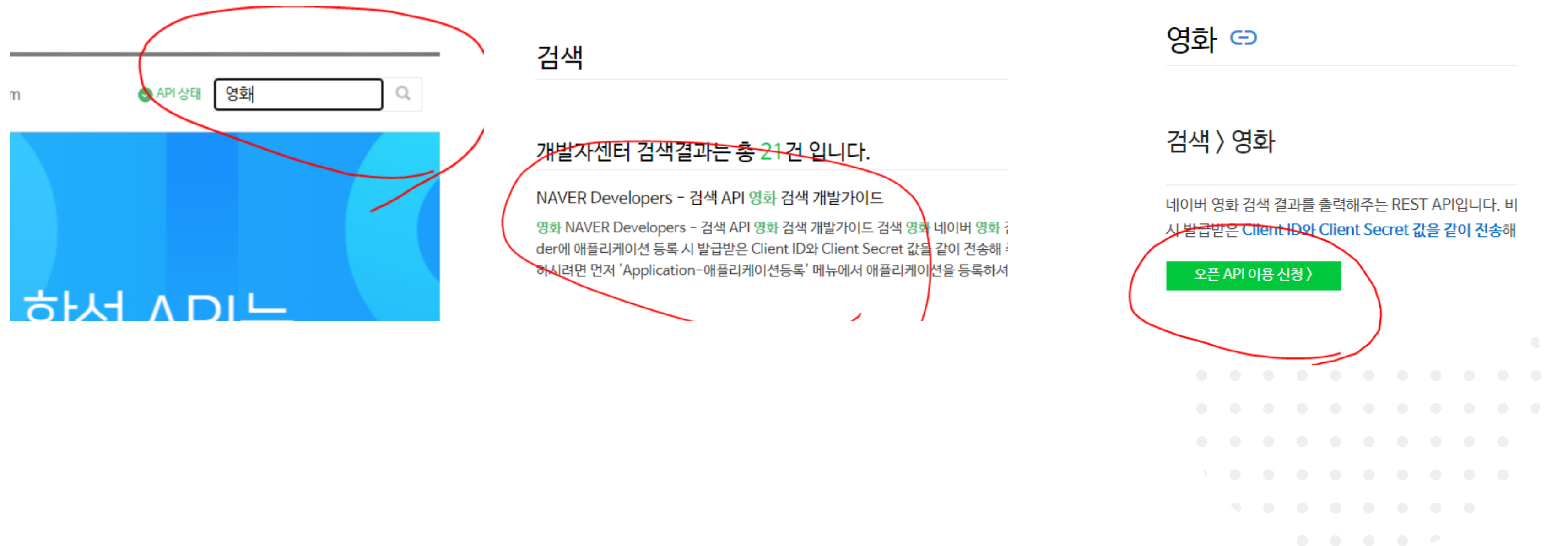
API 키 발급

fetch API 코드 작성

# 네이버 개발자 센터 접속

<https://developers.naver.com/main/>

우측 상단의 검색 창에 영화 검색 - 검색 API 개발 가이드 클릭 - 오픈 API 이용 신청 버튼 클릭



# 네이버 애플리케이션 등록

- 1. 애플리케이션 이름 입력
- 2. 사용 API 검색 등록
- 3. 환경 추가 -> Web
- 4. Web 환경의 웹 서비스 URL에 React 개발 서버 주소 (localhost:3000) 입력
- 5. 등록

애플리케이션의 기본 정보를 등록하면, 좌측 **내 애플리케이션** 메뉴의 서브 메뉴에 등록하신 애플리케이션 이름으로 서브 메뉴가 만들어집니다.

애플리케이션 이름	<div>React 연습</div> <div><ul style="list-style-type: none"><li>• 네이버 로그인할 때 사용자에게 표시되는 이름이므로 서비스 브랜드를 대표할 수 있는 이름으로 가급적 10자 이내로 간결하게 설정해주세요.</li><li>• 40자 이내의 영문, 한글, 숫자, 공백문자, "-", "_" 만 입력 가능합니다.</li></ul></div>
사용 API	<div>선택하세요.</div> <div>검색</div>
비로그인 오픈 API 서비스 환경	<div>환경 추가</div> <div><div>WEB 설정</div><div>웹 서비스 URL (최대 10개)</div><div><div>http://localhost:3000</div><div>+</div></div><div><ul style="list-style-type: none"><li>• 텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '-' 버튼을 누르면 행이 삭제됩니다.</li><li>• http와 https는 구분하지 않습니다.</li><li>• www는 빼고 입력해 주세요. 예) http://naver.com</li><li>• 서브 도메인이 있으면 대표 도메인명만 입력해 주세요. (예: http://naver.com)</li><li>• 하이브리드 앱은 location.href 객체 출력 값을 입력하면 됩니다. (예: file://로컬 URI)</li></ul></div></div>

등록하기

취소

# API 키 발급

Client ID와 Client Secret라는 키를 활용해 API 요청  
이런 API 키는 외부에 노출되면 안되는 정보이므로  
철저히 보안을 지켜 관리

## React 연습

개요	API 설정	멤버관리	로그인 통계
----	--------	------	--------

## 애플리케이션 정보

Client ID	EXZUMG6GFY70U0mpX1bB
Client Secret	<div>.....</div> <div>보기</div>



# API 키 관리

- 프로젝트 루트 폴더(package.json이 있는 위치)에 .env 파일을 생성하고 내용을 다음과 같이 작성

REACT\_APP\_NAVER\_CLIENT\_ID="<Client ID>"

REACT\_APP\_NAVER\_CLIENT\_SECRET="<Client Secret>"

- 그리고 App.js에 console.log(process.env);

작성 후 개발 서버 재 실행

- 이와 같이 .env 파일에 설정, 키 같은 민감한 파일 관리 후 .gitignore에 작성하는 것처럼 저장소에 올리지 않도록 함

```
{NODE_ENV: 'development', PUBLIC_URL: '', WDS_SOCKET_HOST: undefined, WDS_SOCKET_PATH: undefined, WDS_SOCKET_PORT: undefined, ...}
FAST_REFRESH: true
NODE_ENV: "development"
PUBLIC_URL: ""
REACT_APP_NAVER_CLIENT_ID: "<Client ID>"
REACT_APP_NAVER_CLIENT_SECRET: "<Client Secret>"
WDS_SOCKET_HOST: undefined
WDS_SOCKET_PATH: undefined
WDS_SOCKET_PORT: undefined
[[Prototype]]: Object
```

```
.gitignore
1 # See https://help.github.com/
2
3 # dependencies
4 **/node_modules
5 /.pnp
6 .pnp.js
7
8 # testing
9 /coverage
10
11 # production
12 /dist
13 /build
14
15 # misc
16 .DS_Store
17 .env
18 .env.local
19 .env.development.local
20 .env.test.local
```



# CORS 문제 임시 제거

<https://developer.mozilla.org/ko/docs/Web/HTTP/CORS>

- 개발 서버와 같이 클라이언트에서 요청을 하는 경우,  
서버에서 이런 비정상 요청을 막는 경우가 있음

- 그 후, 루트 폴더에 setupProxy.js 파일을 생성하고  
코드 작성 후 개발 서버 재실행

npm install http-proxy-middleware --save-dev

혹은

yarn add -D http-proxy-middleware

명령어로 해당 모듈 설치

※ 임시 방편이므로 실무에서 사용 불가

```
const { createProxyMiddleware } = require("http-proxy-middleware");
module.exports = function (app) {
  app.use(
    "/api",
    createProxyMiddleware({
      target: "https://openapi.naver.com",
      changeOrigin: true,
      pathRewrite: { "^/api/": "/" },
    })
  );
};
```

# API 요청 로직 작성

- /src/apis/fetch.js 파일 생성 후 코드 작성

```
const URL = "/api/v1/search/movie.json?query=어벤져스";  
export const requestGetMovieList = async () => {  
  const response = await fetch(URL, {  
    headers: {  
      "Content-Type": "plain/text",  
      "X-Naver-Client-Id": process.env.REACT_APP_NAVER_CLIENT_ID,  
      "X-Naver-Client-Secret": process.env.REACT_APP_NAVER_CLIENT_SECRET,  
    },  
  });  
  return await response.json();  
};
```

※ API 요청과 같은 로직은 웹 서비스와 관심사가 분리되어 있으므로 폴더를 따로 만들어 모아두는 것이 좋음





# MovieContainer.jsx 작성 후 요청

- /src/containers/MovieContainer.jsx 파일 생성 후 코드 작성

```
import { useCallback, useEffect } from "react";
import { requestGetMovieList } from "../apis/fetch";

const MovieContainer = () => {
  const callApiGetMovieList = useCallback(async () => {
    const data = await requestGetMovieList();
    console.log(data);
  }, []);

  useEffect(() => {
    callApiGetMovieList();
  }, [callApiGetMovieList]);

  return <div></div>;
};

export default MovieContainer;
```

MovieContainer.jsx:7

```
▼ {lastBuildDate: 'Sun, 16 Jan 2022 23:38:52 +0900', total: 11, start: 1, display: 10, items: Array(10)} ⓘ
  display: 10
  ▼ items: Array(10)
    ▶ 0: {title: '레고 마블 <b>어벤져스</b> : 기후위기', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn...
    ▶ 1: {title: '<b>어벤져스</b>: 엔드게임', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=1369...
    ▶ 2: {title: '<b>어벤져스</b>: 인피니티 워', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=1...
    ▶ 3: {title: '<b>어벤져스</b> 오브 저스티스', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=...
    ▶ 4: {title: '<b>어벤져스</b> 그림: 시간 전쟁', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?cod...
    ▶ 5: {title: '<b>어벤져스</b>: 에이지 오브 울트론', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?...
    ▶ 6: {title: '슈퍼히어로 어벤져', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=140241', ima...
    ▶ 7: {title: '<b>어벤져스</b> 컨피덴셜: 블랙 위도우 앤 퍼니셔', link: 'https://movie.naver.com/movie/bi/mi...
    ▶ 8: {title: '디스크 전사 <b>어벤져스</b>', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=14...
    ▶ 9: {title: '<b>어벤져스</b> 어셈블', link: 'https://movie.naver.com/movie/bi/mi/basic.nhn?code=143881'...
    length: 10
    ▶ [[Prototype]]: Array(0)
  lastBuildDate: "Sun, 16 Jan 2022 23:38:52 +0900"
  start: 1
  total: 11
  ▶ [[Prototype]]: Object
>
```

※ 전체 데이터를 가지고 있는 컴포넌트를 일반적으로 컨테이너라고 부름

# 03

## API 요청 시 로딩

useState를 통한 로딩 관리

# 로딩 컴포넌트 작성

- /src/components/Loading.jsx 파일 생성 후 코드 작성

```
import { Backdrop, CircularProgress } from "@mui/material";
```

```
const Loading = ({ isLoading }) => {  
  return (  
    <Backdrop open={isLoading}>  
      <CircularProgress />  
    </Backdrop>  
  );  
};
```

```
export default Loading;
```

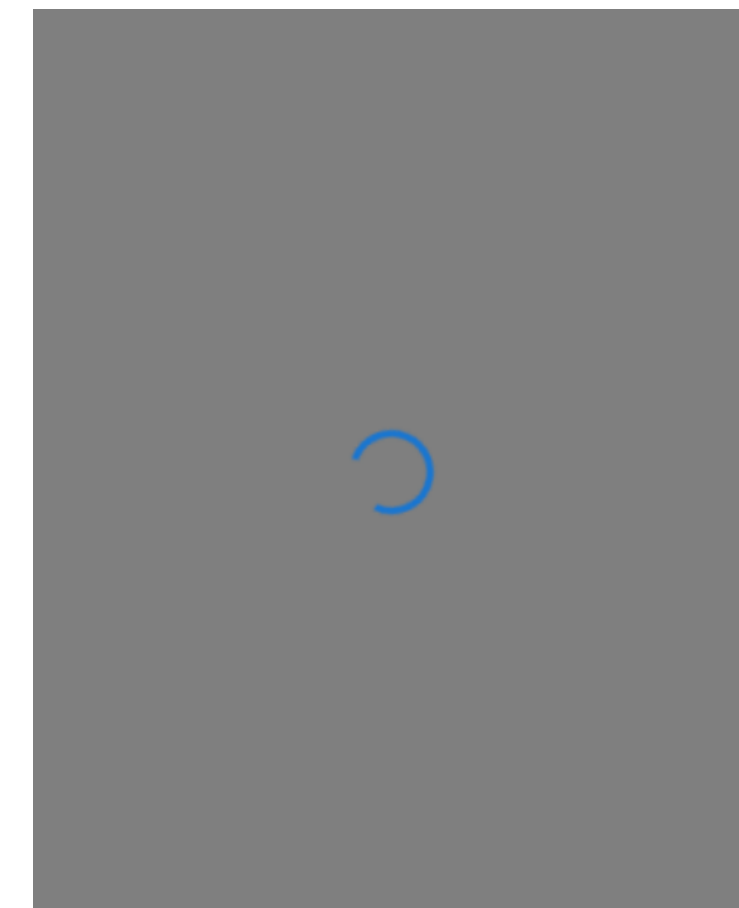


# useState를 통한 로딩 관리

- /src/containers/MovieContainer.jsx 코드 수정

```
import { useState, useCallback, useEffect } from "react";
// ... 중략
const MovieContainer = () => {
  const [loading, setLoading] = useState(false);
  const callApiGetMovieList = useCallback(async () => {
    setLoading(true);
    try {
      const data = await requestGetMovieList();
    } catch(e) {
      console.error(e);
    }
    setLoading(false);
  }, []);
  // ...
```

```
// ...
return (
  <div>
    <Loading isLoading={loading} />
  </div>
);
export default Loading;
```



※ try ... catch로 에러 처리를 하면 해당 함수는 반드시 끝까지 실행되기 때문에 혹여나 에러가 발생하여도 두 번째 setLoading을 실행하지 않아 무한 로딩 상태가 유지되는 버그 발생하지 않음



# 04

## 외부 데이터 렌더링

요청 받은 데이터 state로 관리

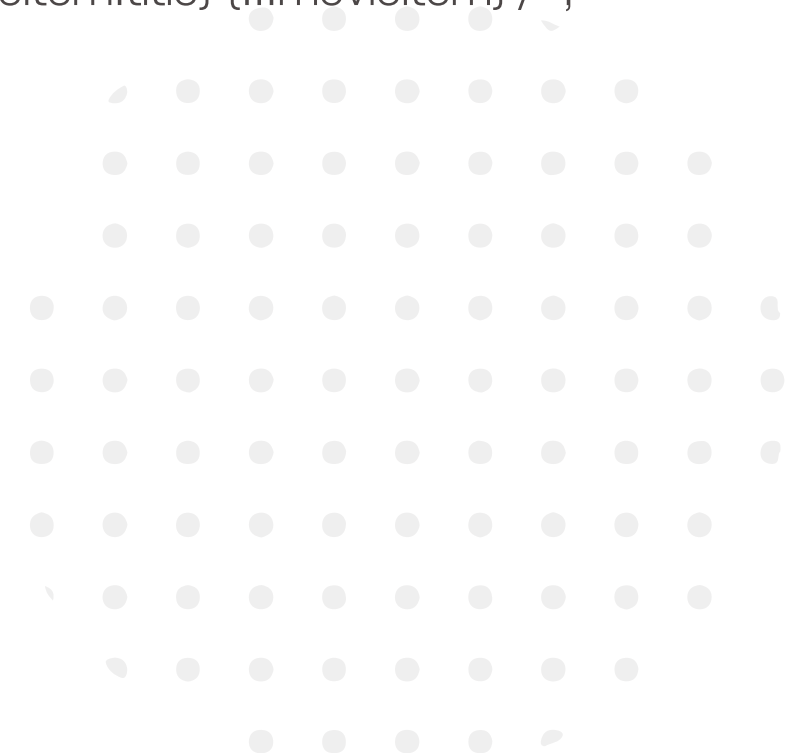
# 요청 받은 데이터 state로 관리

- /src/components/Movie.jsx 파일 생성 후 코드 작성

```
import { memo } from "react";
```

```
const Movie = ({ title, subtitle, image, link, director }) => {  
  return (  
    <div>  
      <a href={link}>  
        <img src={image} alt="movie-thumbnail" />  
        <h3>{title}</h3>  
        <h4>{subtitle}</h4>  
        <p>감독: {director}</p>  
      </a>  
    </div>  
  );  
};
```

```
const MovieList = ({ movieList }) => {  
  return (  
    <div>  
      {movieList.map((movieItem) => {  
        return <Movie key={movieItem.title} {...movieItem} />;  
      })}  
    </div>  
  );  
};  
  
export const Movies = memo(MovieList);  
  
export default Movie;
```



# 요청 받은 데이터 state로 관리

- /src/containers/MovieContainer.jsx 파일에 코드 추가

```
import { Movies } from "../components/Movie";

const MovieContainer = () => {
  const [loading, setLoading] = useState(false);
  const [movieList, setMovieList] = useState([]);
  const callApiGetMovieList = useCallback(async () => {
    setLoading(true);
    try {
      const data = await requestGetMovieList();
      setMovieList(data.items);
    } catch (e) {
      console.error(e);
    }
    setLoading(false);
  }, []);
  // ... 중략
};
```

[레고 마블 <b>어벤져스</b> : 기후위기](#)

[Lego Marvel Avengers: Climate Conundrum](#)

감독: 켄 커닝햄

[<b>어벤져스</b>: 엔드게임](#)

[Avengers: Endgame](#)

감독: 안소니 루소|조 루소

[<b>어벤져스</b>: 인피니티 워](#)

[Avengers: Infinity War](#)

감독: 안소니 루소|조 루소

[<b>어벤져스</b> 오브 저스티스](#)

[Avengers of Justice: Farce Wars](#)

감독:

[<b>어벤져스</b> 그림: 시간 전쟁](#)

[Avengers Grimm: Time Wars](#)

감독: 맥시밀리언 엘펠트

[<b>어벤져스</b>: 에이지 오브 울트론](#)

[The Avengers: Age of Ultron](#)

감독: 조스 웨던

[슈퍼히어로 어벤져](#)

[Avengers Grimm](#)

감독: 제레미 M. 인맨

[<b>어벤져스</b> 컨피덴셜: 블랙 위도우 앤 퍼니셔](#)

[Avengers Confidential: Black Widow & Punisher](#)

감독: 시미즈 켄이치

[디스크 전사 <b>어벤져스</b>](#)

[ディスク &#12539;ウォ &#12540;ズ:アベンジャ &#12540;ズ](#)

감독:

[<b>어벤져스</b> 어셈블](#)

[Avengers Assemble](#)

감독:

실증 아카데미 리액트 특강

THANK  
YOU

(주) 인바이즈 박철현  
(주) 인바이즈 개발팀