

# 영화 앱 만들기



## 1. state로 숫자 증감 기능 만들어 보기

# 1 영화 API 사용해 보기

- ❖ 앞 장에서 영화 데이터를 로딩하려면 자바스크립트의 `fetch()`라는 함수가 필요하다고 했고, `fetch()` 함수 대신 `axios`라는 도구를 사용하겠다고 했다.
- ❖ 우리는 `axios`를 사용해서 영화 앱을 만들 것이다.

## ❖ `axios` 설치하기

- `axios`를 설치하자.
- 터미널에 다음과 같이 입력하면 된다.

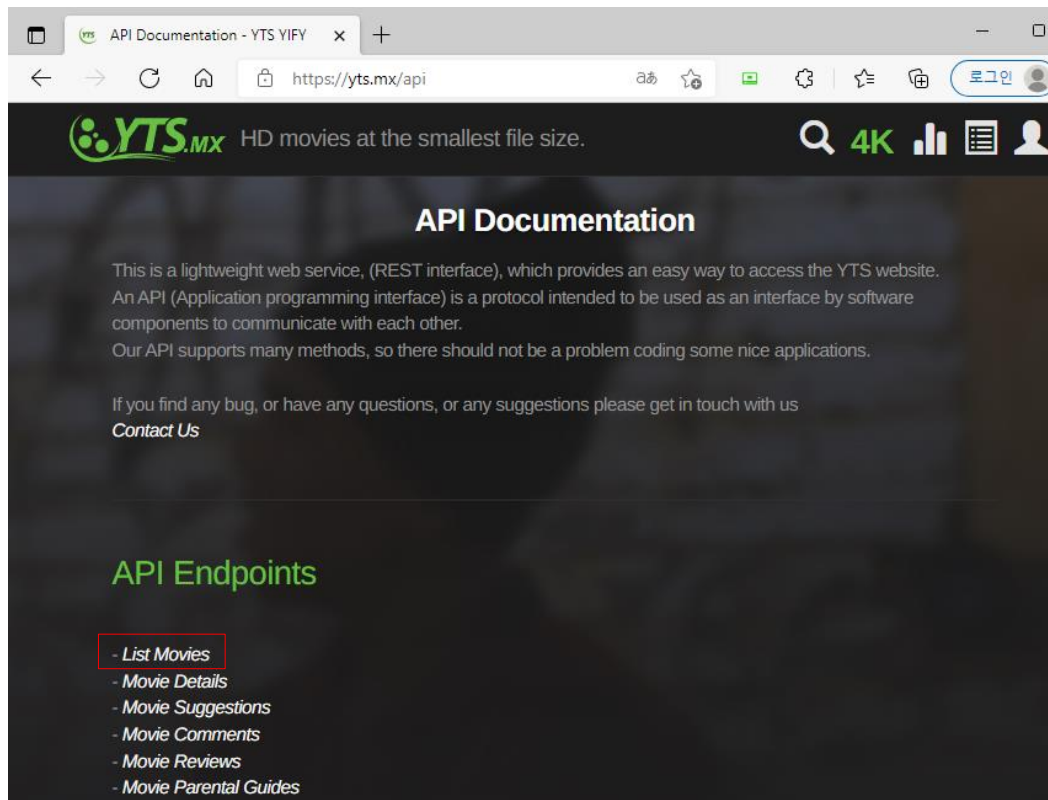
**>npm install axios**

```
C:\movie_app_2021>npm install axios
added 1 package, and audited 1408 packages in 1s
163 packages are looking for funding
  run `npm fund` for details
6 moderate severity vulnerabilities
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
C:\movie_app_2021>_
```

# 1 영화 API 사용해 보기

## ❖ YTS 영화 데이터 API 살펴보기

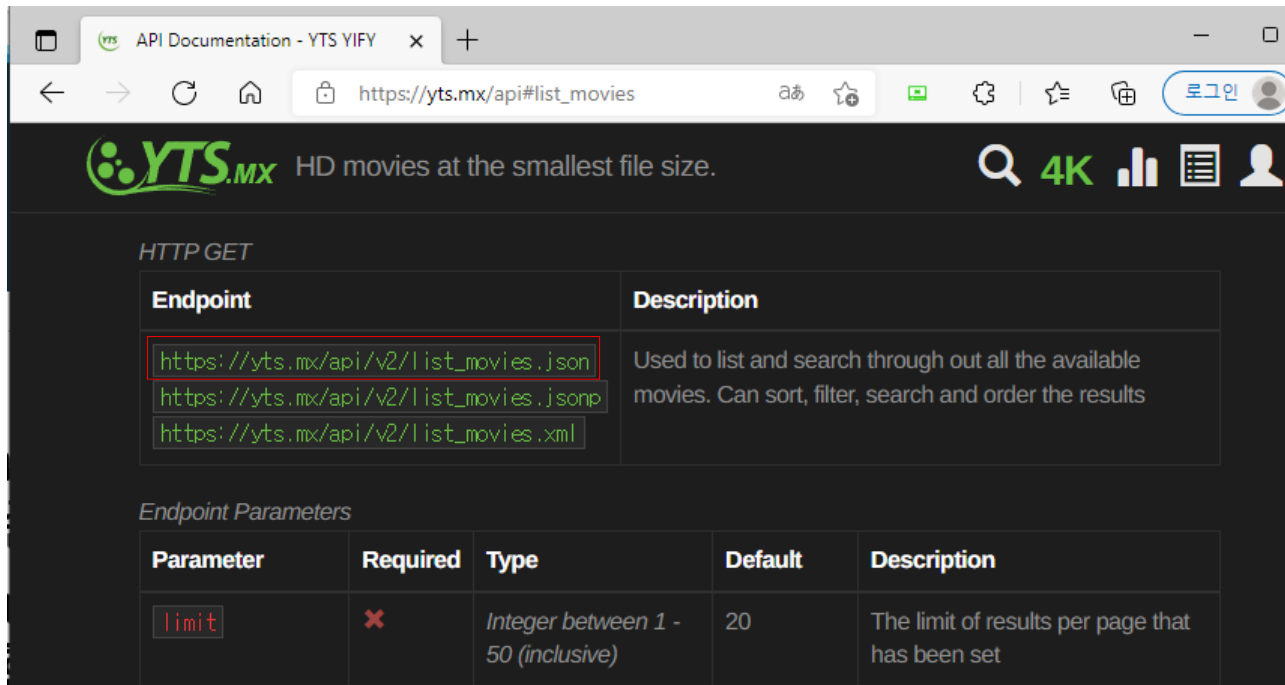
- 브라우저 주소 입력 창에 **yts.lt/api**라고 입력하면 YTS 영화 데이터 API 사이트에 접속할 수 있다.
- 여기서 우리는 'List Movies API'라는 기능을 사용할 것이다.
- <List Movies>를 눌러보자.



# 1 영화 API 사용해 보기

## ❖ YTS 영화 데이터 API 살펴보기

- API는 그림에서 보듯 특정 주소(Endpoint라고 적힌 곳의 주소 참고)를 입력하면 그 주소에 맞는 결과를 보내준다.
- 그리고 추가로 특정 주소에 조건(Endpoint Parameters라고 적힌 곳 참고)을 붙여 입력하면 그 조건까지 고려한 결과를 보내준다.
- 우리는 Endpoint의 가장 위에 있는 주소를 사용할 것이다.
- 이 주소는 최신 영화 20개에 대한 데이터를 기본으로 보내준다.



The screenshot shows a web browser window with the URL `https://yts.mx/api#list_movies`. The page header includes the YTS.MX logo and the tagline "HD movies at the smallest file size." Below the header, the "HTTP GET" section displays a table of endpoints. The first endpoint, `https://yts.mx/api/v2/list_movies.json`, is highlighted with a red box. Below this, the "Endpoint Parameters" section shows a table with parameters for the `list_movies` endpoint.

Endpoint	Description
<code>https://yts.mx/api/v2/list_movies.json</code>	Used to list and search through out all the available movies. Can sort, filter, search and order the results
<code>https://yts.mx/api/v2/list_movies.jsonp</code>	
<code>https://yts.mx/api/v2/list_movies.xml</code>	

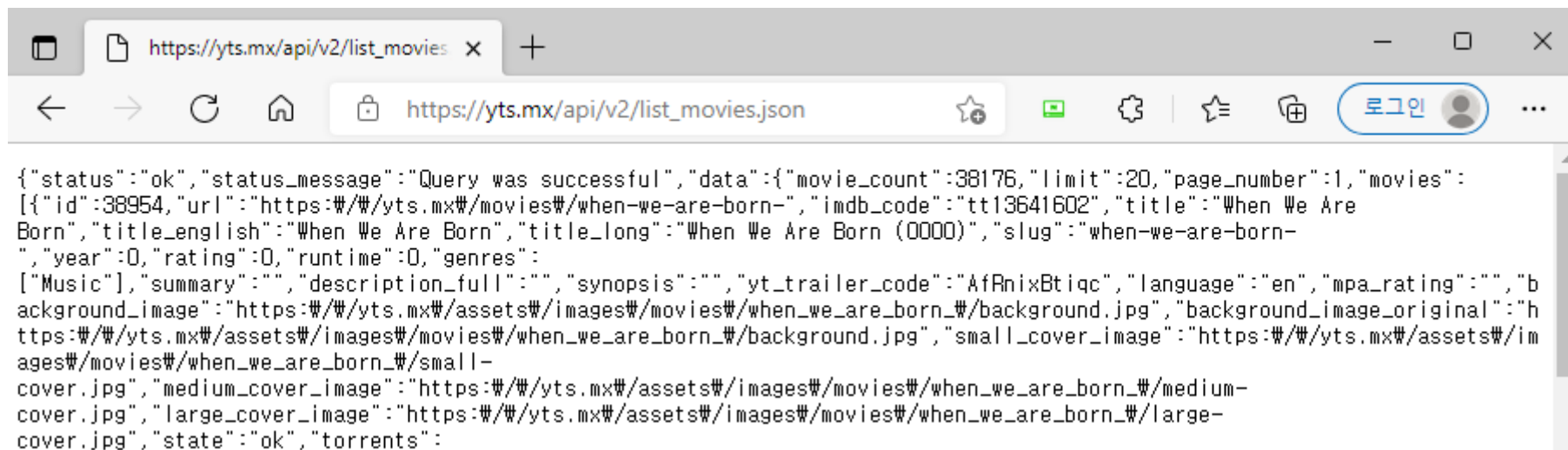
Parameter	Required	Type	Default	Description
<code>limit</code>	✗	Integer between 1 - 50 (inclusive)	20	The limit of results per page that has been set

# 1 영화 API 사용해 보기

## ❖ 영화 목록 데이터 확인해보기

- 정말 그런지 API를 바로 사용해보자.
- 브라우저 주소 창에 있는 Endpoint의 주소 중 .json으로 끝나는 주소를 입력해보자.

**Endpoint 주소: yts.mx/api/v2/list\_movies.json**



# 1 영화 API 사용해 보기

## ❖ 영화 목록 데이터 확인해보기

- 그러면 복잡해 보이는 텍스트가 화면에 표시되는 것을 볼 수 있다.
- 이것이 JSON 데이터이다.
- 지금은 JSON 데이터에 줄바꿈이 없어서 보기가 어렵다.
- JSON 데이터를 좀 더 편하게 보려면 크롬 브라우저의 'JSON Viewer' 라는 확장 도구를 설치하면 된다.

# 1 영화 API 사용해 보기

## ❖ JSON Viewer 확장 도구 설치하기

- 크롬 웹 스토어에서 JSON Viewer를 검색한 다음 <Chrome에 추가>를 눌러 JSON Viewer를 설치하자.



chrome 웹 스토어



jjin300@gm:

홈 > 확장 프로그램 > JSON Viewer



### JSON Viewer

제공자: tulios

★★★★★ 954 | [개발자 도구](#) | 👤 사용자 1,000,000+명

📌 오프라인 실행 가능

Chrome에 추가

개요

개인정보 보호관행

리뷰

지원

관련 프로그램



# 1 영화 API 사용해 보기

## ❖ JSON Viewer 확장 도구 설치하기

- JSON Viewer를 설치한 다음 앞에서 접속했던 주소로 다시 접속하면 JSON 형식의 데이터가 깔끔하게 출력될 것이다.

```
1 // 20220103122526
2 // https://yts.mx/api/v2/list_movies.json
3
4 {
5   "status": "ok",
6   "status_message": "Query was successful",
7   "data": {
8     "movie_count": 38176,
9     "limit": 20,
10    "page_number": 1,
11    "movies": [
12      {
```

응답 상태는 ok

영화 데이터 수는 38176개

영화 데이터는 여기에

# 1 영화 API 사용해 보기

## ❖ JSON Viewer 확장 도구 설치하기

- 여러 정보가 보일 것이다?
- API가 영화 데이터를 출력해 준 것이다.
- 하나씩 살펴보자.
- Status 키값은 응답 상태 메시지이다.
- API의 응답 상태가 정상이므로 "ok"라고 문자열을 보내주고 있다.
- Data 키값에 영화 데이터가 포함되어 있다.
- Movie\_count 키값은 API가 보내준 영화 데이터의 개수고, limit 키값은 보내준 데이터의 개수이다.
- 이런 식으로 API의 데이터를 살펴보면 된다.

# 1 영화 API 사용해 보기

## ❖ JSON Viewer 확장 도구 설치하기

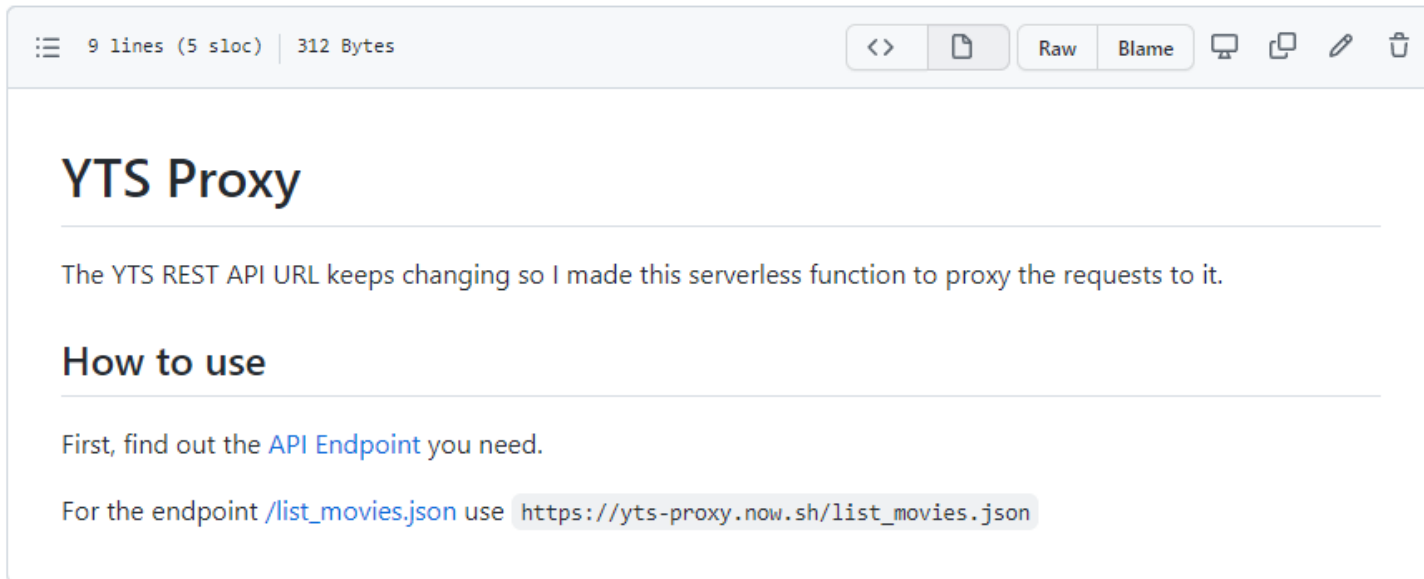
- `movies` 키값이 API에서 보내준 영화 데이터 알맹이이다.
- 화면에서 보듯 `movies`는 배열이고, 그 안에 객체가 들어있다.
- 객체에는 `id`, `url`, `imdb_code`, `title`, ...와 같은 키값이 보인다.
- 살펴보면 우리가 음식 앱에 입력한 데이터와 동일한 구조를 갖는다.
- 음식 앱과 비슷하게 영화 앱을 만들 수 있을 것 같다.
- 그런데 YTS에는 문제가 하나 있다.
- YTS에서 영화 토렌트 파일을 업로드하고 있기 때문이다.
- 이건 불법이다.
- 그러다 보니 매번 접속해야 하는 주소가 변경된다.
- 이후에 주소가 바뀌면 영화 앱을 만들기 곤란하다.
- 그래서 YTS proxy API를 사용하겠다.
- 이건 불법이 아니니까 써도 괜찮다.

# 1 영화 API 사용해 보기

## ❖ 영화 API를 사용하기

- 영화 API 깃허브에 접속해보면 README.md에 간단한 소개 글이 적혀 있다.
- **How to use**를 읽어보자.

<https://github.com/jjin300/ui>



The screenshot shows a GitHub repository page for 'YTS Proxy'. The file is named 'README.md' and has 9 lines (5 sloc) and 312 Bytes. The content of the README is as follows:

```
YTS Proxy
```

---

The YTS REST API URL keeps changing so I made this serverless function to proxy the requests to it.

```
How to use
```

---

First, find out the [API Endpoint](#) you need.

For the endpoint `/list_movies.json` use `https://yts-proxy.now.sh/list_movies.json`

# 1 영화 API 사용해 보기

## ❖ 영화 API를 사용하기

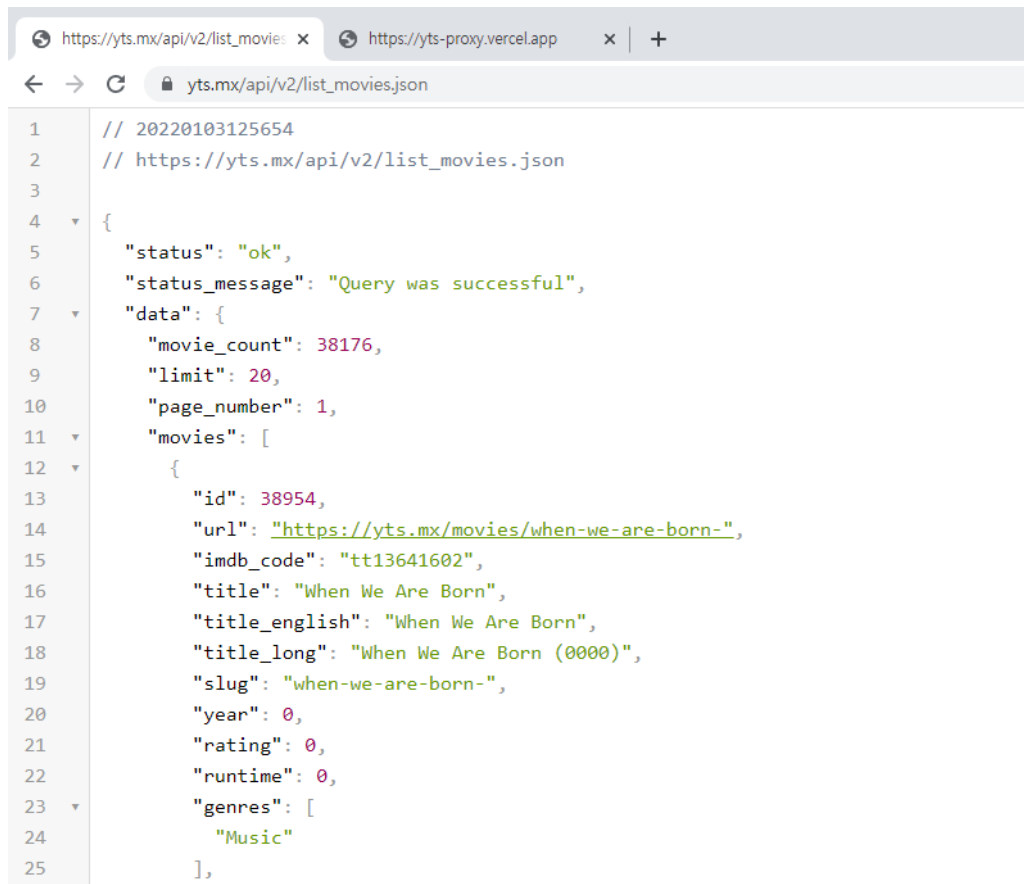
- YTS의 endpoint `/list_movies.json`을 쓰려면 `yts-proxy.now.sh`에 `/list_movies.json`를 붙이면 된다고 설명하고 있다.
- 만약 YTS의 다른 endpoint와 함께 영화 API를 쓰려면 `yts-proxy.now.sh`에 `endpoint#` 붙이기만 하면 된다.
- 다음 표를 보면 영화 API를 어떻게 써야 할지 대충 감이 올 것이다.

endpoint	YTS API	영화 API
list movies	<code>yts.mx/api/v2/list_movies.json</code>	<code>yts-proxy.now.sh/list_movies.json</code>
movie details	<code>yts.mx/api/v2/movie_details.json</code>	<code>yts-proxy.now.sh/movie_details.json</code>

# 1 영화 API 사용해 보기

## ❖ 영화 API를 사용하기

- `yts-proxy.now.sh/list_movies.json`를 입력하면 앞에서 본 것과 같은 결과를 얻을 수 있다.



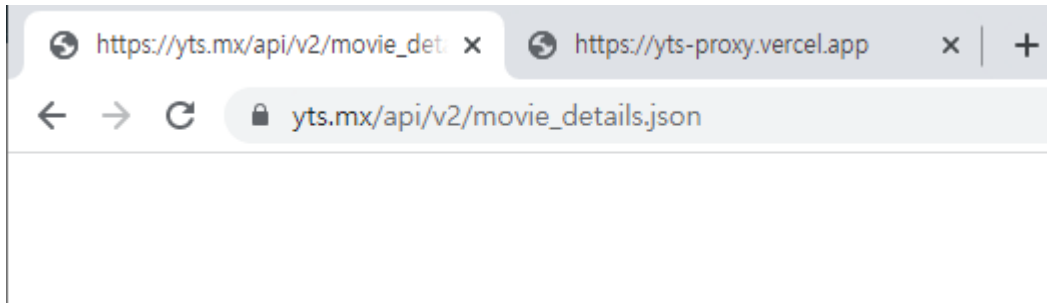
```
1 // 20220103125654
2 // https://yts.mx/api/v2/list_movies.json
3
4 {
5   "status": "ok",
6   "status_message": "Query was successful",
7   "data": {
8     "movie_count": 38176,
9     "limit": 20,
10    "page_number": 1,
11    "movies": [
12      {
13        "id": 38954,
14        "url": "https://yts.mx/movies/when-we-are-born-",
15        "imdb_code": "tt13641602",
16        "title": "When We Are Born",
17        "title_english": "When We Are Born",
18        "title_long": "When We Are Born (0000)",
19        "slug": "when-we-are-born-",
20        "year": 0,
21        "rating": 0,
22        "runtime": 0,
23        "genres": [
24          "Music"
25        ],

```

# 1 영화 API 사용해 보기

## ❖ 영화 정보 더 자세히 살펴보기

- 마지막으로 영화 정보를 더 자세히 보여주는 영화 API를 사용해 보자.
- `yts-proxy.now.sh/movie_details.json`를 입력하면 된다.



- 앗! 그런데 아무것도 안 나온다.
- 왜냐하면 영화 정보를 더 자세히 보여주는 API가 `movie_id`라는 조건을 요구하기 때문이다.

# 1 영화 API 사용해 보기

## ❖ 영화 정보를 더 자세히 보기 위해 조건 추가하기

- `yts.mx/api#movie_details`에 접속하면 `movie_details` Endpoint는 `movie_id`가 필수임을 알 수 있다.
- 즉, `movie_id`를 `yts-proxy.now.sh/movie_details.json`에 추가해야 한다.

HTTP GET

Endpoint	Description
<code>https://yts.mx/api/v2/movie_details.json</code> <code>https://yts.mx/api/v2/movie_details.jsonp</code> <code>https://yts.mx/api/v2/movie_details.xml</code>	Returns the information about a specific movie

Endpoint Parameters

Parameter	Required	Type	Default	Description
<code>movie_id</code>	✓	Integer (Unsigned)	null	The ID of the movie
<code>with_images</code>	✗	Boolean	false	When set the data returned will include the added image URLs
<code>with_cast</code>	✗	Boolean	false	When set the data returned will include the added information about the cast

Examples

URL	Description
<code>https://yts.mx/api/v2/movie_details.json?movie_id=10</code>	Returns basic information about the movie with ID of 10

Movie\_id 추가 예시

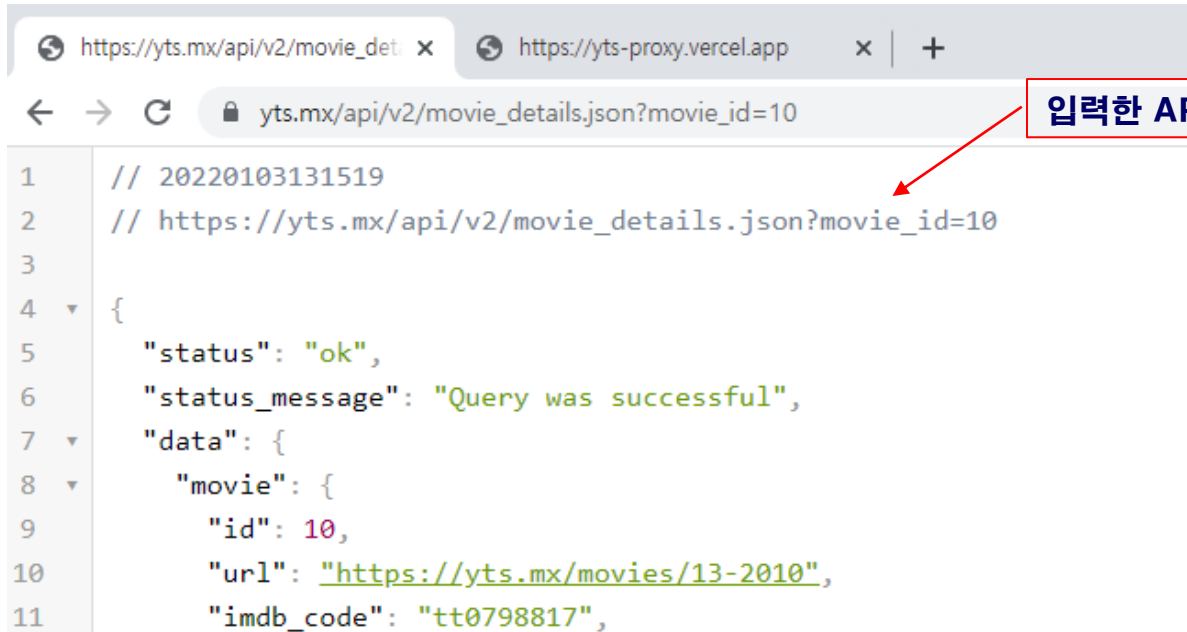


# 1 영화 API 사용해 보기

## ❖ moviejd가 10인 영화 정보 살펴보기

- 우리는 영화 API를 사용할 거니까 `yts-proxy.now.sh`로 시작하는 주소를 쓰면 된다.
- `Yts-proxy.now.sh/movie_details.json`를 입력한 다음 `?movie_id=10`을 이어붙이면 아이디가 10인 영화의 자세한 정보가 나타난다.

[https://yts-proxy.now.sh/movie\\_details.json?movie\\_id=10](https://yts-proxy.now.sh/movie_details.json?movie_id=10)



```
1 // 20220103131519
2 // https://yts.mx/api/v2/movie_details.json?movie_id=10
3
4 {
5   "status": "ok",
6   "status_message": "Query was successful",
7   "data": {
8     "movie": {
9       "id": 10,
10      "url": "https://yts.mx/movies/13-2010",
11      "imdb_code": "tt0798817",
```

입력한 API URL이 여기에 표시된다.

# 1 영화 API 사용해 보기

## ❖ movie\_id가 10인 영화 정보 살펴보기

- 이런 식으로 영화 정보를 하나하나 자세히 가져오면 `year`, `rating`, `runtime`, `genres`, ... 등과 같은 정보도 영화 앱에 출력할 수 있다.
- 그러면 우리의 영화 앱에서 **API**를 사용하려면 어떻게 해야 할까?
- 아주 간단하다.
- `App.js` 파일 맨 위에 `axios`를 `import`한 다음, `componentDidMount()` 함수에서 `axios`로 **API**를 호출하면 된다.

# 1 영화 API 사용해 보기

## ❖ 영화 API를 영화 앱에서 호출하기

- `axios.get()` 함수의 인자에 URL을 전달하여 API를 호출했다.
- 그리고 `setTimeout(...)`은 이제 사용하지 않을 코드니까 지우자.
- `axios` 임포트도 잊지말자!!

1	<code>import React from "react";</code>
2	<code>import axios from "axios";</code>
3	
4	<code>class App extends React.Component{</code>
5	<code>  state = {</code>
6	<code>    isLoading: true,</code>
7	<code>    movies: [],</code>
8	<code>  };</code>
9	
10	<code>  componentDidMount(){</code>
11	<code>    axios.get('https://yts-proxy.now.sh/list_movies.json');</code>
12	<code>  }</code>
13	

# 1 영화 API 사용해 보기

## ❖ 영화 API를 영화 앱에서 호출하기

- `axios.get()` 함수의 인자에 URL을 전달하여 API를 호출했다.
- 그리고 `setTimeout(...)`은 이제 사용하지 않을 코드니까 지우자.
- `axios` 임포트도 잊지말자!!

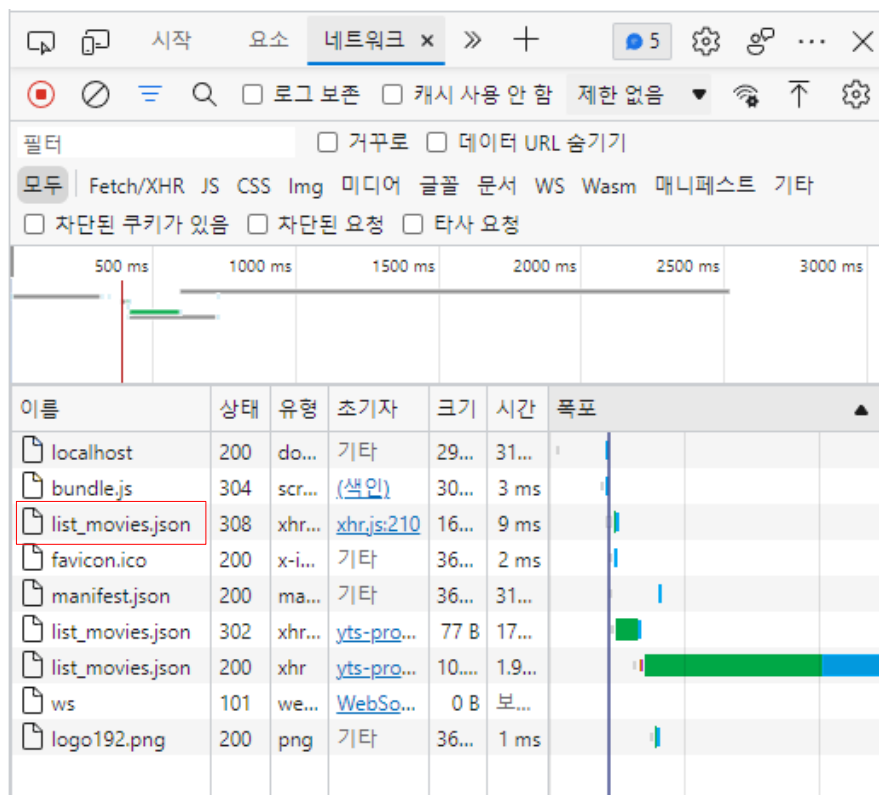
14	<code>render(){</code>
15	<code>  const {isLoading} = this.state;</code>
16	<code>  return &lt;div&gt;{isLoading ? 'Loading...' : 'We are ready'}&lt;/div&gt;;</code>
17	<code>  }</code>
18	<code>}</code>
19	
20	<code>export default App;</code>

- 영화 앱을 실행해 보면 여전히 `Loading...` 이라고만 나온다.
- 하지만 중요한 건 `axios`가 동작하고 있다는 것이다.
- 오류가 발생하지 않으니까.
- 즉, `axios`는 분명 API에 영화 데이터를 요청하고 있다.
- 어떻게 그 사실을 알 수 있을까?

# 1 영화 API 사용해 보기

## ❖ Axios의 동작 확인해 보기

- [네트워크] 탭을 연 다음 영화 앱을 새로 고침하자.
- 그러면 [네트워크] 탭의 내용 중 Name이라는 항목에 `list_movies.json`이라고 나온 부분이 있다.
- 이건 axios가 API를 호출하고 있기 때문에 생긴 것이다.



이름	상태	유형	초기자	크기	시간	폭포
localhost	200	do...	기타	29...	31...	
bundle.js	304	scr...	(색인)	30...	3 ms	
list_movies.json	308	xhr...	xhr.js:210	16...	9 ms	
favicon.ico	200	x-i...	기타	36...	2 ms	
manifest.json	200	ma...	기타	36...	31...	
list_movies.json	302	xhr...	yts-pro...	77 B	17...	
list_movies.json	200	xhr	yts-pro...	10....	1.9...	
ws	101	we...	WebSo...	0 B	보...	
logo192.png	200	png	기타	36...	1 ms	

# 1 영화 API 사용해 보기

## ❖ Axios의 동작 확인해 보기

- [네트워크] 탭을 보니 `axios`로 얻은 데이터로 아무것도 하지 않고 있을 뿐이지 `axios`는 분명 잘 동작하고 있는 것 같다.
- 혹시 [콘솔] 탭에 오류가 나타났다면 이후 내용을 진행하면 자연스럽게 고쳐질 거니까 걱정하지 않아도 된다.
- 아무튼 이제 `axios`를 활용한 API 호출에 성공했으니 우리에게 필요한 영화 데이터를 추출하면 된다.
- 여기서 잠깐! `axios`는 네트워크를 사용하므로 느리게 동작한다.
- 그래서 `axios.get()`이 반환한 영화 데이터를 잡으려면 자바스크립트에게 `axios.get()`을 포함하고 있는 함수의 실행이 끝날 때까지 시간이 걸릴 수 있다고 말해야 한다.
- 그러니 `axios.get()`의 실행이 분리될 수 있도록 새 함수를 만들자.

# 1 영화 API 사용해 보기

❖ **getMovies()** 함수 기다린 다음, **axios.get()** 함수가 반환한 데이터 잡기

- **getMovies()** 함수를 만들고, 그 함수 안에서 **axios.get()**이 실행되도록 만들자.
- **axios.get()**이 반환한 값은 **movies**에 저장했다.

	...
4	class App extends React.Component{
	...
10	<b>getMovies = () =&gt; {</b>
11	<b>const movies = axios.get("https://yts-proxy.now.sh/list_movies.json");</b>
12	<b>}</b>
13	
14	componentDidMount(){
15	<b>this.getMovies();</b>
16	}
	...

# 1 영화 API 사용해 보기

❖ `getMovies()` 함수 기다린 다음, `axios.get()` 함수가 반환한 데이터 잡기

- 이제 `componentDidMount()` 함수가 실행되면 `this.getMovies()`가 실행된다.
- 이때 자바스크립트에게 '`getMovies()` 함수는 시간이 좀 필요해' 라고 말해야만 `axios.get()`이 반환한 데이터를 제대로 잡을 수 있다.
- 그렇게 하려면 두 가지 키워드가 필요하다.
- 그게 바로 `async`, `await`이다.



# 1 영화 API 사용해 보기

## ❖ `getMovies()`에 `async` 붙이고, `axios.get()`에 `await` 붙이기

- 자바스크립트에게 `getMovies()` 함수는 시간이 필요해'라고 말하려면 `async`를 () 앞에 붙이고 실제 시간이 필요한 대상인 `axios.get()` 앞에 `await` 를 붙이면 된다.
- `movies` 에 있는 값이 궁금하면 `console.log()`로 출력해 봐도 좋다.

	...
4	class App extends React.Component{
	...
10	getMovies = <b>async</b> () => {
11	const movies = <b>await</b> axios.get("https://yts-proxy.now.sh/list_movies.json");
12	}
13	
14	componentDidMount(){
15	this.getMovies();
16	}
	...

자바스크립트에게 `getMovies()` 함수는 시간이 필요하고

`axios.get()`의 실행을 기다려 달라고 알려준다.

# 1 영화 API 사용해 보기

## ❖ `getMovies()`에 `async` 붙이고, `axios.get()`에 `await` 붙이기

- 사실 `async`라는 키워드는 자바스크립트에게 `getMovies()` 함수가 비동기라고 말해준다.
- 자바스크립트에게 `getMovies()` 함수는 비동기라서 기다려야 해'라고 자바스크립트에게 말한 거라고 생각하면 된다.
- `await`라는 키워드는 자바스크립트에게 `getMovies()` 함수 내부의 `axios.get()`의 실행 완료를 기다렸다가 끝나면 계속 진행해 줘'라고 말해준다.
- 여기서 우리가 집중해야 할 내용은 'API에서 데이터를 받아오는 `axios.get()`을 실행하려면 시간이 필요하고, 그 사실을 자바스크립트에게 알려야만 데이터를 잡을 수 있으므로 `async`, `await`를 사용했다'는 점이다.
- 만약 `async`, `await`가 무엇인지 궁금하다면 자바스크립트를 공부해야 한다.
- 앱을 실행해보면 아직 큰 변화가 없다.
- 하지만 바로 다음 내용에서 `axios.get()`으로 잡은 영화 데이터를 출력해 볼 것이다.

# 1 영화 API 사용해 보기

## ❖ 지금까지 만든 영화 앱의 실행 시나리오

- 지금까지 만든 영화 앱의 실행 시나리오를 복습해보자.
- 리액트 앱이 실행되면 최초로 `render( )` 함수가 실행된다.
- 최초의 `state`에는 `isLoading`, `movies`가 있다.
- `isLoading`는 `true`이고 `movies`는 빈 배열이다.
- 그러다보니 최초의 실행 화면에는 `Loading...`이 표시된다.
- 이어서 `App` 컴포넌트가 마운트되면 `componentDidMount( )` 함수가 실행되면서 `getMovies( )` 함수가 실행된다.
- 그런데 `getMovies( )` 함수에는 시간이 많이 걸리는(느린) `axios.get( )`이 포함되어 있다.
- 그래서 `getMovies()` 함수에 `async`를, `axios.get()`에 `await`를 붙였다.
- 이때 `async`는 `await`와 짝꿍이다.
- 그래서 둘은 동시에 사용해야 한다는 것을 기억하자.

## 2 영화 데이터 화면에 그리기

### ❖ `console.log()` 함수로 영화 데이터 출력해 보기

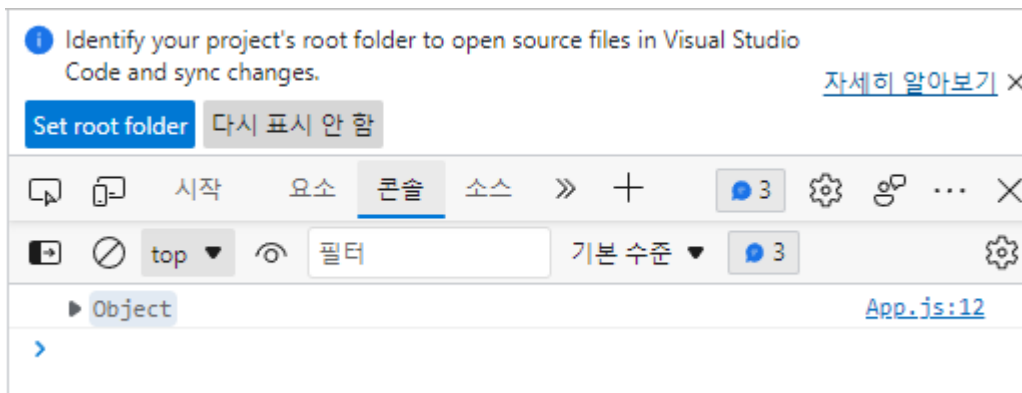
- `axios.get()`으로 잡은 영화 데이터가 `movies` 변수 안에 들어 있으니 `console.log(movies)`를 통해 출력해 보자.

	...
4	<code>class App extends React.Component{</code>
	...
10	<code>  getMovies = async() =&gt; {</code>
11	<code>    const movies = await axios.get("https://yts-proxy.now.sh/list_movies.json");</code>
12	<code>    console.log(movies);</code>
13	<code>  }</code>
14	
15	<code>  componentDidMount(){</code>
16	<code>    this.getMovies();</code>
17	<code>  }</code>
	...

## 2 영화 데이터 화면에 그리기

### ❖ `console.log()` 함수로 영화 데이터 출력해 보기

- `axios.get()`으로 잡은 영화 데이터가 `movies` 변수 안에 들어 있으니 `console.log(movies)`를 통해 출력해 보자.

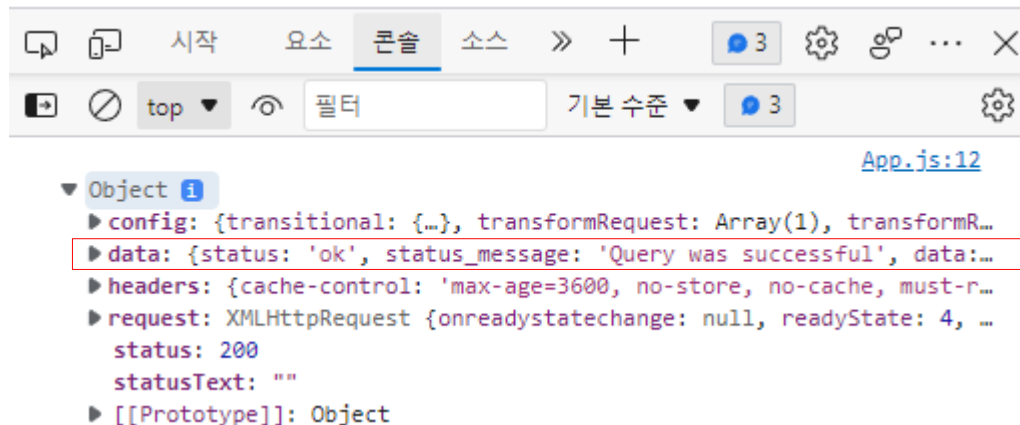


- [콘솔] 탭을 보면 `object`와 같이 짧게 표현된 객체가 있다.
- 이것이 API로 받아온 영화 데이터이다.
- 한번 확인해보자.

## 2 영화 데이터 화면에 그리기

### ❖ 영화 데이터 자세히 살펴보기

- 왼쪽에 있는 ▶ 표시를 눌러서 펼치면, 객체를 자세히 표시해준다.



- 우리는 이 중에서 **data** 키에 집중할 것이다.
- 거기에 우리에게 필요한 영화 데이터가 들어있기 때문이다.

## 2 영화 데이터 화면에 그리기

### ❖ 영화 데이터 자세히 살펴보기

- **data** 키를 또 펼치면 그 안에 또 다시 **data** 키가 들어있다.
- 자세히 살펴보면 그 안에 **movies** 배열이 있다.
- **movies** 배열까지 펼쳐 보자.

```
▼ Object 1
  ▶ config: {transitional: {...}, transformRequest: Array(1), transformR...
  ▼ data:
    ▶ @meta: {server_time: 1641191672, server_timezone: 'CET', api_ver...
    ▼ data:
      limit: 20
      movie_count: 38176
      ▼ movies: Array(20)
        ▶ 0: {id: 38954, url: 'https://yts.mx/movies/when-we-are-born-...
        ▶ 1: {id: 38952, url: 'https://yts.mx/movies/my-love-2021', im...
        ▶ 2: {id: 38951, url: 'https://yts.mx/movies/wicked-duchess-19...
        ▶ 3: {id: 38948, url: 'https://yts.mx/movies/frankenstein-90-1...
        ▶ 4: {id: 38949, url: 'https://yts.mx/movies/911-fifteen-years...
        ▶ 5: {id: 38950, url: 'https://yts.mx/movies/tell-her-that-i-l...
        ▶ 6: {id: 38947, url: 'https://yts.mx/movies/cinderellas-hot-n...
        ▶ 7: {id: 38946, url: 'https://yts.mx/movies/bikini-model-mayh...
        ▶ 8: {id: 38945, url: 'https://yts.mx/movies/cute-devil-2018',...
        ▶ 9: {id: 38944, url: 'https://yts.mx/movies/dernier-atout-194...
        ▶ 10: {id: 38941, url: 'https://yts.mx/movies/shake-hands-with...
        ▶ 11: {id: 38939, url: 'https://yts.mx/movies/mansion-of-the-d...
        ▶ 12: {id: 38938, url: 'https://yts.mx/movies/come-what-may-20...
        ▶ 13: {id: 38937, url: 'https://yts.mx/movies/earth-emergency-...
        ▶ 14: {id: 38935, url: 'https://yts.mx/movies/loreal-paris-wom...
        ▶ 15: {id: 38934, url: 'https://yts.mx/movies/the-murder-of-ga...
        ▶ 16: {id: 38933, url: 'https://yts.mx/movies/right-to-try-202...
        ▶ 17: {id: 38932, url: 'https://yts.mx/movies/the-housewives-o...
        ▶ 18: {id: 38931, url: 'https://yts.mx/movies/days-of-our-live...
        ▶ 19: {id: 38930, url: 'https://yts.mx/movies/christmas-propos...
        length: 20
      ▶ [[Prototype]]: Array(0)
      page_number: 1
      ▶ [[Prototype]]: Object
```

## 2 영화 데이터 화면에 그리기

### ❖ 영화 데이터 자세히 살펴보기

- **movies** 키 안에 우리에게 필요한 영화 데이터가 들어있다.
- **id**(아이디), **url**(주소), **imdb\_code**, **title**(제목), **title\_english**(영어 제목)와 같은 키가 들어있다.
- **data** → **data** → **movies** 순서대로 객체에 접근하면 원하는 데이터를 추출할 수 있다.



## 2 영화 데이터 화면에 그리기

### ❖ 객체에 있는 `movies` 키에 접근하기

- 그러면 정말로 원하는 데이터를 추출하기 위해 `movies` 변수에 있는 `movies` 키의 값을 추출해 보자.
- `movies.data.data.movies`와 같이 수정한 다음 [콘솔] 탭을 확인해 보자.

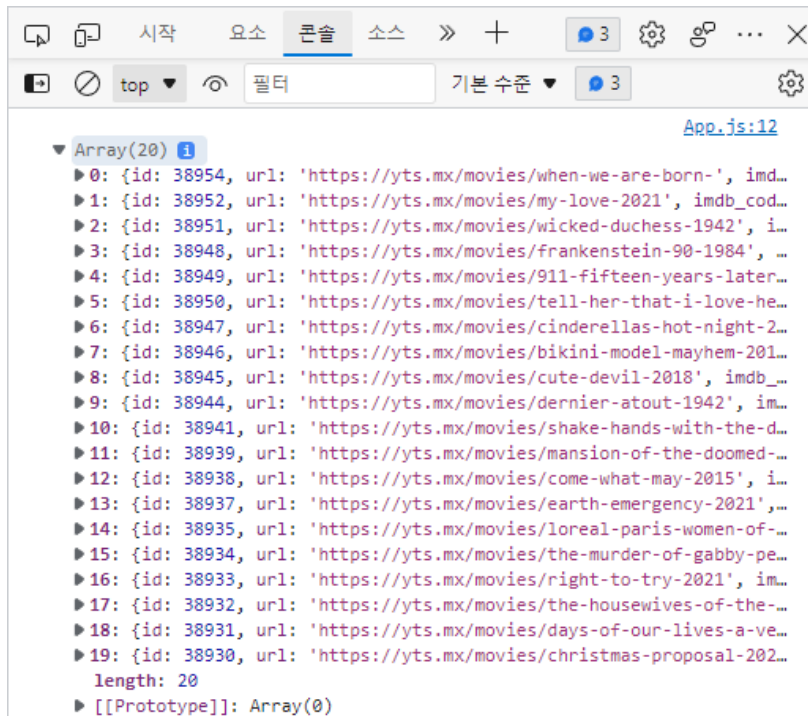
	...
4	class App extends React.Component{
	...
10	getMovies = async() => {
11	const movies = await axios.get("https://yts-proxy.now.sh/list_movies.json");
12	console.log(movies.data.data.movies);
13	}
14	
15	componentDidMount(){
16	this.getMovies();
17	}
	...

점 연산자로 객체에 접근하면 된다.

## 2 영화 데이터 화면에 그리기

### ❖ 객체에 있는 **movies** 키에 접근하기

- **movies** 변수에는 이제 배열이 들어있다.
- 배열에는 객체가 20개 들어 있다.
- 이제 우리가 원하는 진짜 영화 데이터만 추출하게 되었다.
- 이 데이터의 구조는 음식 앱을 만들 때 사용했던 데이터 구조와 같다.
- 그러니 화면에 출력하는 것도 어렵지 않다.



```
App.js:12
▼ Array(20)
  ▶ 0: {id: 38954, url: 'https://yts.mx/movies/when-we-are-born-', imd...
  ▶ 1: {id: 38952, url: 'https://yts.mx/movies/my-love-2021', imdb_cod...
  ▶ 2: {id: 38951, url: 'https://yts.mx/movies/wicked-duchess-1942', i...
  ▶ 3: {id: 38948, url: 'https://yts.mx/movies/frankenstein-90-1984', ...
  ▶ 4: {id: 38949, url: 'https://yts.mx/movies/911-fifteen-years-later...
  ▶ 5: {id: 38950, url: 'https://yts.mx/movies/tell-her-that-i-love-he...
  ▶ 6: {id: 38947, url: 'https://yts.mx/movies/cinderellas-hot-night-2...
  ▶ 7: {id: 38946, url: 'https://yts.mx/movies/bikini-model-mayhem-201...
  ▶ 8: {id: 38945, url: 'https://yts.mx/movies/cute-devil-2018', imdb_...
  ▶ 9: {id: 38944, url: 'https://yts.mx/movies/dernier-atout-1942', im...
  ▶ 10: {id: 38941, url: 'https://yts.mx/movies/shake-hands-with-the-d...
  ▶ 11: {id: 38939, url: 'https://yts.mx/movies/mansion-of-the-doomed-...
  ▶ 12: {id: 38938, url: 'https://yts.mx/movies/come-what-may-2015', i...
  ▶ 13: {id: 38937, url: 'https://yts.mx/movies/earth-emergency-2021',...
  ▶ 14: {id: 38935, url: 'https://yts.mx/movies/loreal-paris-women-of-...
  ▶ 15: {id: 38934, url: 'https://yts.mx/movies/the-murder-of-gabby-pe...
  ▶ 16: {id: 38933, url: 'https://yts.mx/movies/right-to-try-2021', im...
  ▶ 17: {id: 38932, url: 'https://yts.mx/movies/the-housewives-of-the-...
  ▶ 18: {id: 38931, url: 'https://yts.mx/movies/days-of-our-lives-a-ve...
  ▶ 19: {id: 38930, url: 'https://yts.mx/movies/christmas-proposal-202...
    length: 20
  ▶ [[Prototype]]: Array(0)
```

## 2 영화 데이터 화면에 그리기

### ❖ 객체에 있는 `movies` 키에 조금 더 똑똑하게 접근하기

- 다만! `movies.data.data.movies`와 같은 방법으로 객체에 접근하는 건 아름답지 않다.
- ES6를 사용한다면 구조 분해 할당을 사용하는 게 좋다.
- 다음과 같이 구조 분해 할당을 사용하도록 코드를 수정하여 `movies` 키에 접근해보자.

	...
4	class App extends React.Component{
	...
10	getMovies = async() => {
11	const {
12	data: {
13	movies, ← 여기서 data → movies가 진행된다.
14	},
15	} = await axios.get("https://yts-proxy.now.sh/list_movies.json");
16	console.log(movies);
17	};
	...

## 2 영화 데이터 화면에 그리기

### ❖ movies state에 영화 데이터 저장하기

- `this.setState({ movies : movies })`와 같이 작성하면 `movies state`에 영화 데이터를 저장할 수 있다.
- `Console.log()`는 이제 사용하지 않을 거니까 지우자.

	...
4	<code>class App extends React.Component{</code>
	...
10	<code>getMovies = async() =&gt; {</code>
11	<code>  const {</code>
12	<code>    data:{</code>
13	<code>      data: {movies},</code>
14	<code>    },</code>
15	<code>  } = await axios.get("https://yts-proxy.now.sh/list_movies.json");</code>
16	<code>this.setState({movies: movies});</code>
17	<code>};</code>
	...
	<div>이건 state</div>
	<div>이건 구조 분해 할당으로 얻은 영화 데이터가 있는 변수이다.</div>

## 2 영화 데이터 화면에 그리기

### ❖ `movies state`에 영화 데이터 저장하기

- 참고로 `movies state`와 `axios.get()`의 결과를 저장할 변수 `movies`의 이름이 같다고 해서 둘을 혼동하면 안된다.
- 하나는 `state`고 하나는 `axios.get()`의 결과를 담을 변수이다.
- ES6에서는 객체의 키와 대입할 변수의 이름이 같다면 코드를 축약할 수 있다.
- `{movies : movies}`는 키와 대입할 변수의 이름이 같으니까 `{movies}`로 축약할 수 있다.
- `this.setState({movies : movies})`를 `this.setState({movies})`로 수정하자.

## 2 영화 데이터 화면에 그리기

### ❖ movies state에 영화 데이터 저장하기

	...
4	class App extends React.Component{
	...
10	getMovies = async() => {
11	const {
12	data:{
13	data: {movies},
14	},
15	} = await axios.get("https://yts-proxy.now.sh/list_movies.json");
16	this.setState({movies});
17	};
	...

축약 형태로 코드를 바꾸자

- 영화 앱을 실행하면 여전히 Loading...이 나오고 있다.
- We are ready가 출력되려면 어떻게 해야 할까?
- isLoading state의 값을 true에서 false로 업데이트하면 된다.

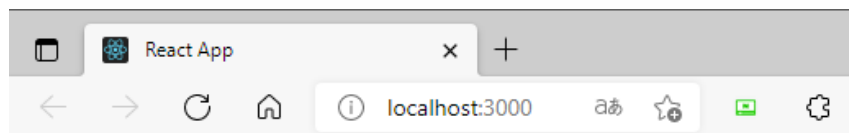
## 2 영화 데이터 화면에 그리기

❖ **isLoading state true에서 false로 업데이트하기**

- **isLoading state도 false에서 true로 업데이트하자.**

	...
4	class App extends React.Component{
	...
10	getMovies = async() => {
11	const {
12	data:{
13	data: {movies},
14	},
15	} = await axios.get("https://yts-proxy.now.sh/list_movies.json");
16	this.setState({movies, isLoading: false});
17	};
	...

movies, isLoading state를 변경했다



We are ready

## 2 영화 데이터 화면에 그리기

### ❖ `isLoading state true`에서 `false`로 업데이트하기

- 영화 앱을 실행해 보면 처음에는 `isLoading...`이 화면에 나타나다가, 조금 뒤에 `We are ready`로 바뀌는 것을 볼 수 있다.
- 영화 데이터를 가져오는 데 성공했고, 로딩 상태 변경까지 성공했다.
- 다만! 아직도 영화 데이터를 출력하고 있지 않다.
- 단순히 문자열 `We are ready`를 출력하고 있다.
- 이건 우리가 바라던 결과가 아니다.
- `Movies state`를 화면에 그려야 한다.
- 그러기 위해서 `Movie` 컴포넌트가 필요하다.



# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트 만들기

- src 폴더에 **Movie.js** 파일을 새로 만들고 다음과 같이 **Movie** 컴포넌트의 기본 골격을 작성해 보자.
- 아직은 출력할 내용이 없으므로 `<h1></h1>`을 반환하도록 만들었다.

1	<code>import React from "react";</code>
2	<code>import propTypes from "prop-types";</code>
3	
4	<code>function Movie(){</code>
5	<code>  return &lt;h1&gt;&lt;/h1&gt;;</code>
6	<code>}</code>
7	
8	<code>Movie.propTypes = {};</code>
9	
10	<code>export default App;</code>

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트 만들기

- **Movie** 컴포넌트는 **state**가 필요하지 않으므로 클래스형 컴포넌트가 아니라 함수형 컴포넌트로 작성한다.
- 또, **Movie**에 넘겨와야 하는 영화 데이터를 정의하고 관리하기 위해 **prop-types**를 사용했다.
- 가장 중요한 건 영화 데이터이다.
- 그러니까 **Movie.propTypes**의 내용을 우선 채우자.
- **Movie.propTypes**의 내용을 채우기 위해 영화 **API**로 받은 데이터를 다시 한번 살펴보자.

# 3 Movie 컴포넌트 만들기

## ❖ 영화 데이터 다시 살펴보기

- [콘솔] 탭에 출력된 내용은 확인하기 불편하니까 `yts-proxy.now.sh/list_movies.json`에 접속해서 우리가 사용할 영화 데이터를 다시 확인해 보자.

```
1 // 20220103202814
2 // https://yts.mx/api/v2/list_movies.json
3
4 {
5   "status": "ok",
6   "status_message": "Query was successful",
7   "data": {
8     "movie_count": 38177,
9     "limit": 20,
10    "page_number": 1,
11    "movies": [
12      {
13        "id": 38954,
14        "url": "https://yts.mx/movies/when-we-are-born-",
15        "imdb_code": "tt13641602",
16        "title": "When We Are Born",
17        "title_english": "When We Are Born",
18        "title_long": "When We Are Born (0000)",
19        "slug": "when-we-are-born-",
20        "year": 0,
21        "rating": 0,
22        "runtime": 0,
23        "genres": [
24          "Music"
25        ],

```

id와

title과

rating 등등 필요한 데이터를 고르자

# 3 Movie 컴포넌트 만들기

## ❖ `Movie.propTypes` 작성하기

- 우선 `id`를 `Movie.propTypes`에 추가해 보자.

1	<code>import React from "react";</code>
2	<code>import propTypes from "prop-types";</code>
3	
4	<code>function Movie(){</code>
5	<code>  return &lt;h1&gt;&lt;/h1&gt;;</code>
6	<code>}</code>
7	
8	<code>Movie.propTypes = {id: propTypes.number.isRequired};</code>
9	
10	<code>export default App;</code>

- `id`는 자료형이 `Number`이고, 반드시 있어야 하니까 `PropTypes.number.isRequired`로 작성했다.
- 이제 나머지 `Movie.propTypes`도 추가해보자.

# 3 Movie 컴포넌트 만들기

## ❖ Movie.propTypes 작성하기

- year, title, summary, poster를 각각 Movie.propTypes에 추가하자.
- 자료형에 주의해서 추가하자.

	...
8	Movie.propTypes = {
9	id: propTypes.number.isRequired,
10	year: propTypes.number.isRequired,
11	title: propTypes.string.isRequired,
12	summary: propTypes.string.isRequired,
13	poster: propTypes.string.isRequired,
14	};
15	
16	export default App;

API에서 넘어오는 id는 숫자

year도 숫자

title, summary는 문자열

poster에는 이미지 주소(문자열)가 저장

- 여기서 하나 주목해야 할 props가 있다.
- poster props는 영화 포스터 이미지 주소를 저장하기 위한 것이다.

# 3 Movie 컴포넌트 만들기

## ❖ Movie.propTypes 작성하기

- `yts-proxy.now.sh/list_movies.json`에 접속한 다음 스크롤을 조금만 내리면 `medium_cover_image` 키를 찾을 수 있다.
- 키와 키값을 자세히 살펴보자.

```
"summary": "",
"description_full": "",
"synopsis": "",
"yt_trailer_code": "AfRnixBtiqc",
"language": "en",
"mpa_rating": "",
"background_image": "https://yts.mx/assets/images/movies/when_we_are_born_/background.jpg",
"background_image_original": "https://yts.mx/assets/images/movies/when_we_are_born_/background.jpg",
"small_cover_image": "https://yts.mx/assets/images/movies/when_we_are_born_/small-cover.jpg",
"medium_cover_image": "https://yts.mx/assets/images/movies/when_we_are_born_/medium-cover.jpg",
"large_cover_image": "https://yts.mx/assets/images/movies/when_we_are_born_/large-cover.jpg",
"state": "ok",
"torrents": [
```

# 3 Movie 컴포넌트 만들기

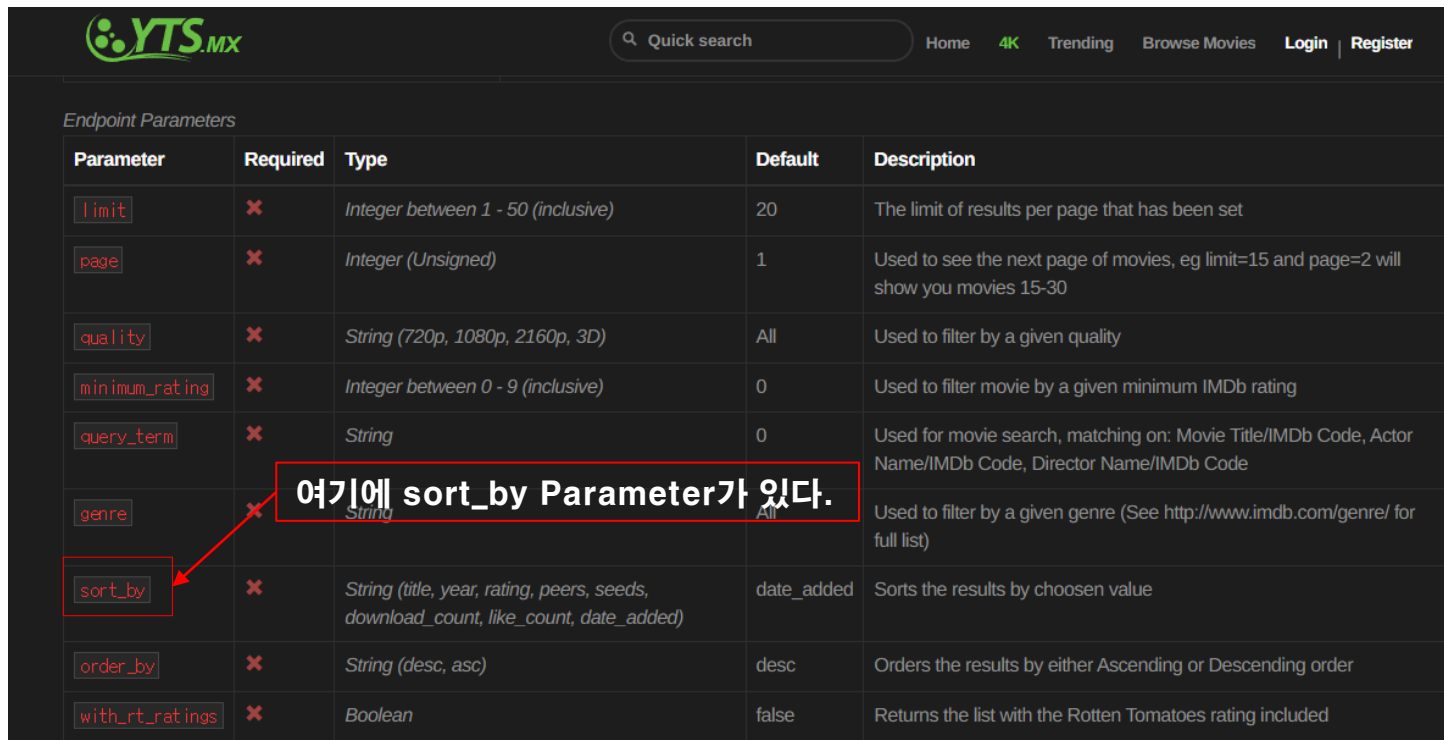
## ❖ `Movie.propTypes` 작성하기

- `medium_cover_image` 키값에 영화 포스터 이미지가 저장되어 있는 주소가 있다.
- 이 값을 사용하면 영화 포스터 이미지도 쉽게 출력할 수 있다.
- 다만 `props`의 이름을 이해하기 쉽도록 API에서 정해진 `medium cover image`가 아니라 `poster`라고 지정했다.
- 이제 `Movie`에 필요한 `prop-types`를 다 추가했다.
- 그런데 우리가 만들 영화 앱이 그냥 영화 포스터와 정보를 출력해 준다면 매력이 떨어질 것 같다.
- 평점(`rating`) 순서로 정렬해서 보여주는 영화 앱이면 매력이 올라가겠지?
- 그렇게 하려면 어떻게 해야 할까?

# 3 Movie 컴포넌트 만들기

## ❖ 영화 API 정렬 기능 사용해 보기

- 바로 영화 API에 구현되어 있는 정렬 기능을 사용하면 된다.
- API 문서를 한번 살펴보자.
- Yts.lt/api#list\_movies에 접속한 다음 Endpoint Parameters를 주목해보자.
- sort\_by라는 Parameter가 보일 것이다.



Endpoint Parameters

Parameter	Required	Type	Default	Description
limit	✗	Integer between 1 - 50 (inclusive)	20	The limit of results per page that has been set
page	✗	Integer (Unsigned)	1	Used to see the next page of movies, eg limit=15 and page=2 will show you movies 15-30
quality	✗	String (720p, 1080p, 2160p, 3D)	All	Used to filter by a given quality
minimum_rating	✗	Integer between 0 - 9 (inclusive)	0	Used to filter movie by a given minimum IMDb rating
query_term	✗	String	0	Used for movie search, matching on: Movie Title/IMDb Code, Actor Name/IMDb Code, Director Name/IMDb Code
genre	✗	String	All	Used to filter by a given genre (See <a href="http://www.imdb.com/genre/">http://www.imdb.com/genre/</a> for full list)
sort_by	✗	String (title, year, rating, peers, seeds, download_count, like_count, date_added)	date_added	Sorts the results by choosen value
order_by	✗	String (desc, asc)	desc	Orders the results by either Ascending or Descending order
with_rt_ratings	✗	Boolean	false	Returns the list with the Rotten Tomatoes rating included



# 3 Movie 컴포넌트 만들기

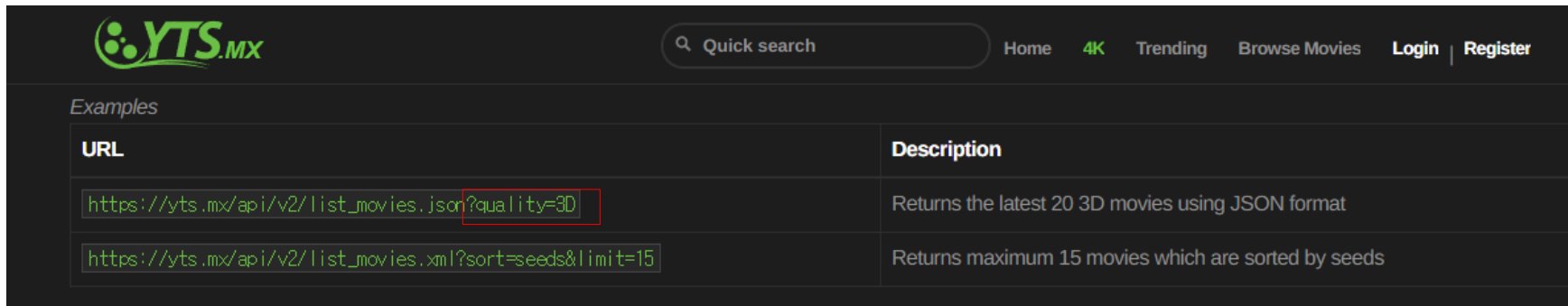
## ❖ 영화 API 정렬 기능 사용해 보기

- `Parameter`의 이름에서 볼 수 있듯 `sort_by`를 사용하면 영화 데이터를 정렬할 수 있다.
- 그러면 `Parameter`를 사용하려면 어떻게 해야 할까?
- 문서에서 제공하는 `Examples`을 보면 된다.

# 3 Movie 컴포넌트 만들기

## ❖ 영화 API 정렬 기능 사용해 보기

- Examples를 보자.
- Example에는 `quality`가 3D인 영화만 불러오는 URL이 적혀 있다.
- ? 오른쪽에 무엇이 적혀 있나 보자.
- `Parameter(quality)`와 `Parameter`에 넘겨줄 값(3D)을 =으로 이어서 작성하면 된다.
- 이런 식으로 `sort_by`를 사용하면 된다.



The screenshot shows the YTS.MX website interface. At the top, there is a navigation bar with the YTS.MX logo, a search bar labeled 'Quick search', and links for 'Home', '4K', 'Trending', 'Browse Movies', 'Login', and 'Register'. Below the navigation bar, there is a section titled 'Examples' which contains a table with two columns: 'URL' and 'Description'.

URL	Description
<code>https://yts.mx/api/v2/list_movies.json?quality=3D</code>	Returns the latest 20 3D movies using JSON format
<code>https://yts.mx/api/v2/list_movies.xml?sort=seeds&amp;limit=15</code>	Returns maximum 15 movies which are sorted by seeds

# 3 Movie 컴포넌트 만들기

## ❖ 영화 API 정렬 기능 사용해 보기

- `yts-proxy.now.sh/list_movies.json?sort_by=rating`에 접속해보자.
- 그러면 평점 내림차순으로 영화데이터를 보여준다.
- 그 증거로 첫 번째 영화는 `rating` 키의 키값이 9.4이다.
- 스크롤바를 쭉 내리면서 살펴보면 정말 평점 내림차순으로 정렬된 것을 알 수 있다.

# 3 Movie 컴포넌트 만들기

## ❖ axios.get() 수정하기

- 이제 새로운 URL로 영화 데이터를 불러와야 하니까 axios.get()에 yts-proxy.now.sh/list\_movies.json?sort\_by=rating을 전달하자.
- App.js

	...
4	class App extends React.Component{
	...
10	getMovies = async() => {
11	const {
12	data:{
13	data: {movies},
14	},
15	} = await axios.get("https://yts-proxy.now.sh/list_movies.json?sort_by=rating");
16	this.setState({movies: movies});
17	};
	...

# 3 Movie 컴포넌트 만들기

## ❖ `axios.get()` 수정하기

- 이제 평점 내림차순으로 영화 데이터를 가져올 수 있게 되었으므로 App 컴포넌트에서 Movie 컴포넌트로 `id`, `title`, `year`, `summary`, `poster` props를 넘겨주면 되겠지?
- Movie 컴포넌트가 이 props를 받아 출력할 수 있도록 Movie 컴포넌트를 마저 완성해 보자.

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트에 props 추가하고 출력해 보기

- **Movie** 컴포넌트에서 **id**, **title**, **year**, **summary**, **poster** props를 받아 출력할 수 있도록 수정하자.
- 그리고 **App** 컴포넌트에서 **Movie** 컴포넌트를 그릴 때 **title**만 우선 출력하도록 만들어 보자.

	...
4	function Movie({id, title, year, summary, poster}){
5	return <h4>{title}</h4>;
6	}
7	
8	Movie.propTypes = {
	...

- 자! 이제 **Movie** 컴포넌트를 작성했다.
- **App** 컴포넌트에서 **Movie** 컴포넌트를 그리면 **title**이 출력되도록 만들자.
- 음식 앱을 만들 때 컴포넌트를 **map()** 함수로 출력했던 것과 같은 방식으로 코딩하면 영화 제목이 주르륵 출력될 것이다.

# 3 Movie 컴포넌트 만들기

## ❖ App 컴포넌트에서 Movie 컴포넌트 그리기

- 구조 분해 할당으로 `this.state`에 있는 `movies`를 얻은 다음, App 컴포넌트에서 `We are ready`를 출력하고 있는 자리(로딩이 다 되면 실행되는 자리)에 `movies.map()`을 사용하자.

	...
23	render(){
24	const {isLoading, <b>movies</b> } = this.state;
25	return <div>{isLoading ? 'Loading...' : <b>movies.map()</b> }</div>;
26	}
27	}
28	
29	export default App;

- 음식 앱을 통해 `map()` 함수를 공부했으니까, `movies.map()`을 어떻게 사용해야 하는지 이미 알고 있을 것이다.
- `Map()` 함수의 첫 번째 인자로 컴포넌트를 반환하는 함수를 전달하면 됐던 거 기억하는가?

# 3 Movie 컴포넌트 만들기

## ❖ map() 함수에 컴포넌트를 반환하는 함수 전달하기

- 코드를 조금씩 전개하여 완성하기 위해 우선 [콘솔] 탭에 영화 데이터를 출력한 다음, 아무것도 반환하지 않는 함수를 전달해 보자.

	...
23	render(){
24	const {isLoading, movies} = this.state;
25	return(
26	<div>
27	{isLoading
28	? 'Loading...'
29	: movies.map((movie) => {
30	console.log(movie);
31	return;
32	})}
33	</div>
34	);
35	}
36	}
37	
38	export default App;

movies는 배열이고, 배열의 원소 1개가 movie로 넘어온다.

여기에서 Movie 컴포넌트를 반환하면 된다.



# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트 반환하도록 `movies.map()` 수정하기

- 이제 **Movie** 컴포넌트를 반환하도록 수정하자.
- **Movie** 컴포넌트를 임포트한 다음, `movies.map()`에 전달한 함수가 `<Movie />`를 반환하도록 만들면 돼.

1	<code>import React from "react";</code>	
2	<code>import axios from "axios";</code>	
3	<code>import Movie from "./Movie";</code>	← Movie 컴포넌트를 임포트하고
	<code>...</code>	
24	<code>render(){</code>	
	<code>...</code>	
31	<code>console.log(movie);</code>	
32	<code>return &lt;Movie/&gt;;</code>	← 여기서 Movie 컴포넌트를 출력한다.
	<code>...</code>	

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트에 props 전달하기

- 이제 Movie 컴포넌트에 props를 전달하자.
- 아까 Movie 컴포넌트에서 id, year, title, summary, poster를 `isRequired`로 설정했다.
- 그러니 설정한 props를 모두 전달해야 한다.
- 대부분 props의 이름은 영화 API에서 받아온 키 이름과 똑같이 만들었다.
- 단! Poster props의 경우 키 이름이 `medium_cover_image`이므로 `movies.poster`가 아니라 `movies.medium_cover_image` 라고 작성해야 한다.

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트에 props 전달하기

### ■ App.js

	...
24	render(){
	...
31	console.log(movie);
32	return(
	<Movie
31	id={movie.id}
32	year={movie.year}
33	title={movie.title}
34	summary={movie.summary}
35	poster={movie.medium_cover_image}
36	/>
37	);
38	}}}
39	</div>
	...

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트에 props 전달하기

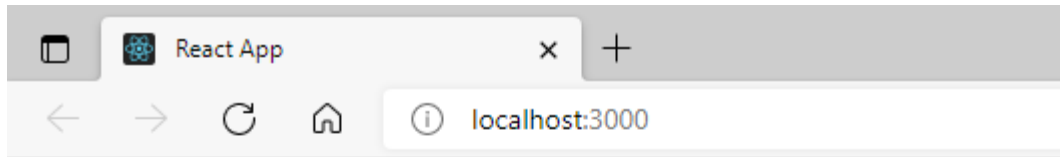
### ■ App.js

	...
24	render(){
	...
31	console.log(movie);
32	return(
	<Movie
31	id={movie.id}
32	year={movie.year}
33	title={movie.title}
34	summary={movie.summary}
35	poster={movie.medium_cover_image}
36	/>
37	);
38	}}}
39	</div>
	...

# 3 Movie 컴포넌트 만들기

## ❖ Movie 컴포넌트에 props 전달하기

- 영화 앱을 실행해 보면 영화 제목이 출력되는 것을 확인할 수 있다.



Jai Bhim

Doctor Who The Day of the Doctor

Days of Reckoning: The Making of US4

The Shawshank Redemption

Avenged Sevenfold: Live in the L.B.C. & Diamonds in the Rough

The Godfather

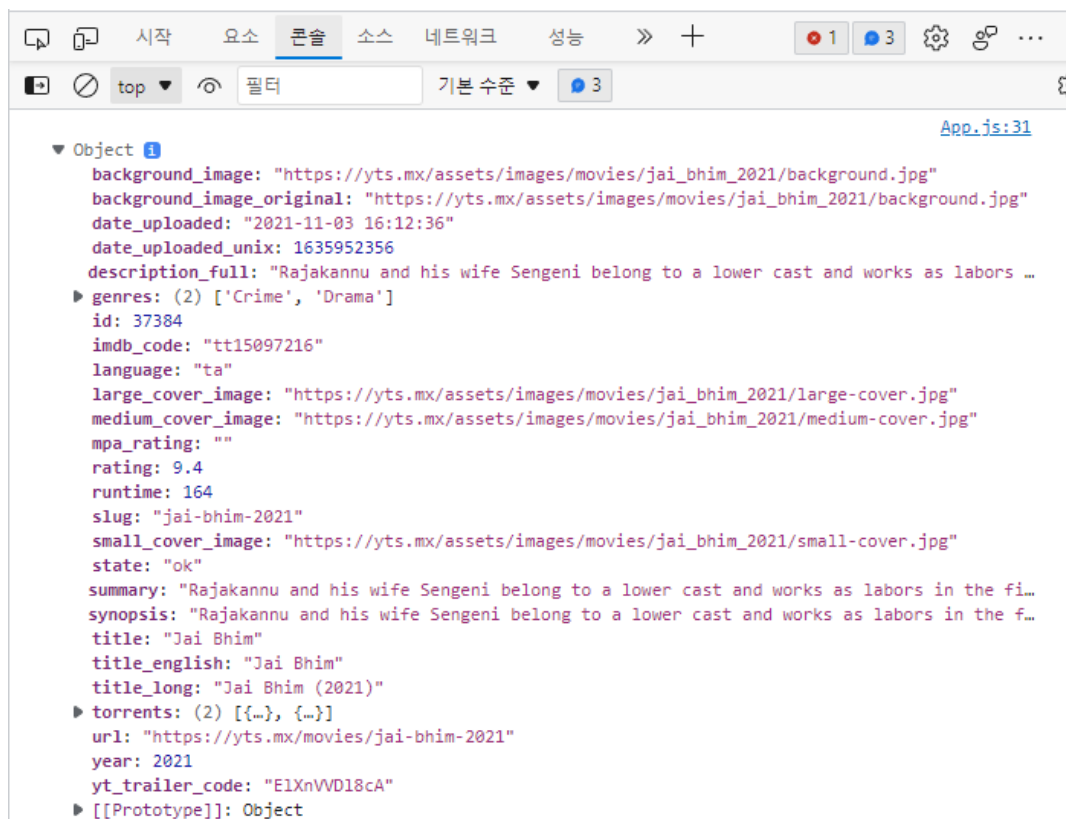
My Beautiful Stutter

Top Gear Africa Special, Part 1

# 3 Movie 컴포넌트 만들기

## ❖ [콘솔] 탭에서 영화 데이터 확인해 보기

- [콘솔] 탭을 보면 `console.log()`가 출력해 주는 내용을 확인할 수 있다.
- 영화 정보가 잘들어오고 있는가?
- 그것도 평점순으로!



```
▼ Object 1
  background_image: "https://yts.mx/assets/images/movies/jai_bhim_2021/background.jpg"
  background_image_original: "https://yts.mx/assets/images/movies/jai_bhim_2021/background.jpg"
  date_uploaded: "2021-11-03 16:12:36"
  date_uploaded_unix: 1635952356
  description_full: "Rajakannu and his wife Sengeni belong to a lower cast and works as labors ..."
  genres: (2) ['Crime', 'Drama']
  id: 37384
  imdb_code: "tt15097216"
  language: "ta"
  large_cover_image: "https://yts.mx/assets/images/movies/jai_bhim_2021/large-cover.jpg"
  medium_cover_image: "https://yts.mx/assets/images/movies/jai_bhim_2021/medium-cover.jpg"
  mpa_rating: ""
  rating: 9.4
  runtime: 164
  slug: "jai-bhim-2021"
  small_cover_image: "https://yts.mx/assets/images/movies/jai_bhim_2021/small-cover.jpg"
  state: "ok"
  summary: "Rajakannu and his wife Sengeni belong to a lower cast and works as labors in the fi..."
  synopsis: "Rajakannu and his wife Sengeni belong to a lower cast and works as labors in the f..."
  title: "Jai Bhim"
  title_english: "Jai Bhim"
  title_long: "Jai Bhim (2021)"
  torrents: (2) [{...}, {...}]
  url: "https://yts.mx/movies/jai-bhim-2021"
  year: 2021
  yt_trailer_code: "ElXnVVD18cA"
  [[Prototype]]: Object
```

# 3 Movie 컴포넌트 만들기

## ❖ [콘솔] 탭에서 영화 데이터 확인해 보기

- 그나저나 [콘솔] 탭에 **key props** 경고 메시지가 나오고 있다.
- 이것도 말끔하게 해결해 보자.

# 3 Movie 컴포넌트 만들기

## ❖ key props 추가하기

- 다음과 같이 코드를 수정하여 **key props** 문제를 해결하자.
- **Key props**는 유일해야 하므로 영화 **API**에서 넘겨주는 **id**를 그대로 사용했다.
- **Console.log()**는 사용하지 않을거니까 지우자.

	...
24	render(){
25	const {isLoading, movies} = this.state;
26	return(
27	<div>
28	{isLoading
29	? 'Loading...'
30	: movies.map((movie) => {
31	
32	return(
33	<Movie
34	key={movie.id} ← 여기에 key props를 추가하면 된다.
35	id={movie.id}



# 3 Movie 컴포넌트 만들기

## ❖ key props 추가하기

- 다음과 같이 코드를 수정하여 **key props** 문제를 해결하자.
- **Key props**는 유일해야 하므로 영화 **API**에서 넘겨주는 **id**를 그대로 사용했다.
- **Console.log()**는 사용하지 않을거니까 지우자.

36	year={movie.year}
37	title={movie.title}
38	summary={movie.summary}
39	poster={movie.medium_cover_image}
40	/>
41	);
42	}}}
43	</div>
44	);
45	}
46	}
47	
48	export default App;

# 3 Movie 컴포넌트 만들기

## ❖ key props 추가하기

- 다시 영화 앱을 실행하면 **key props**와 관련된 경고 메시지가 사라졌음을 확인할 수 있다.
- 지금까지 만든 내용을 정리해보자.
- 우리는 영화 **API**를 통해 영화 데이터를 가져왔다.
- 영화 데이터를 가져올 때는 **axios.get()** 함수를 사용했고, **axios.get()**은 시간이 필요한 함수이므로 **async**, **await**를 사용했다.
- 그리고 **state**에 영화 데이터가 저장되면(업데이트되면) **isLoading...**을 보여주던 화면을 **Movie** 컴포넌트를 보여주도록 만들었다.
- 다음에는 영화 앱을 예쁘게 꾸며 보고, 나머지 **props**도 출력하도록 만들어 보자.

## 4 영화 앱 스타일링 하기 - 기초

- ❖ 지금까지 열심히 영화 앱을 만들었으니 이제는 예쁘게 꾸밀 차례이다.
- ❖ 스타일링의 핵심은 *CSS*지만 *CSS*를 적용할 *HTML*도 필요하다.
- ❖ 그러니까 우선 *HTML*을 작성해 보자.
- ❖ *App* 컴포넌트, *Movie* 컴포넌트에 *HTML*을 추가할 거야.

## 4 영화 앱 스타일링 하기 - 기초

### ❖ App 컴포넌트에 HTML 추가하기

- App 컴포넌트가 반환할 JSX의 바깥쪽을 (section class='container') </section>로, Loading...은 <div class="loader"><span class="loader\_\_text"></span></div>으로, movies.map()은 <div class='movies"></div> 로 감싸자.
- 열고 닫는 태그 표현을 할 때 주의해서 작성하자.

	...
24	render(){
25	const {isLoading, movies} = this.state;
26	return(
27	<section class="container">
28	{isLoading ? (
29	<div class="loader">
30	<span class="loader__text">Loading...</span>
31	</div>
32	): (
33	<div class="movies">
34	{movies.map(movie => (
35	<Movie

JSX의 가장 바깥쪽은 section 엘리먼트로 감쌌다.

여기는 Loading...을 위한 것이다.

Movie 컴포넌트들은 이 엘리먼트로 감쌀 것이다.

## 4 영화 앱 스타일링 하기 - 기초

### ❖ App 컴포넌트에 HTML 추가하기

- App 컴포넌트가 반환할 JSX의 바깥쪽을 (section class='container') </section>로, Loading...은 <div

```
36         key={movie.id}
37         id={movie.id}
38         year={movie.year}
39         title={movie.title}
40         summary={movie.summary}
41         poster={movie.medium_cover_image}
42     />
43     )})
44 </div>
45 })
46 </section>
47 );
48 }
49 }
50
51 export default App;
```

## 4 영화 앱 스타일링 하기 - 기초

### ❖ Movie 컴포넌트에 HTML 추가하기

- Movie 컴포넌트에도 HTML을 추가해 보자.
- Movie 컴포넌트가 반환할 JSX를 `<div class="movie"></div>`로 감싸고, 그 안에서 `title`, `year`, `summary`를 목적에 맞는 엘리먼트로 감싸자.

1	<code>import React from "react";</code>
2	<code>import propTypes from "prop-types";</code>
3	
4	<code>function Movie(){</code>
5	<code>  return(</code>
6	<code>    &lt;div class="movie_data"&gt;</code>
7	<code>      &lt;h3 class="movie_title"&gt;{title}&lt;/h3&gt;</code>
8	<code>      &lt;h5 class="movie_year"&gt;{year}&lt;/h5&gt;</code>
9	<code>      &lt;p class="movie_summary"&gt;{summary}&lt;/p&gt;</code>
10	<code>    &lt;/div&gt;</code>
11	<code>  );</code>
12	<code>}</code>
13	

## 4 영화 앱 스타일링 하기 - 기초

### ❖ Movie 컴포넌트에 HTML 추가하기

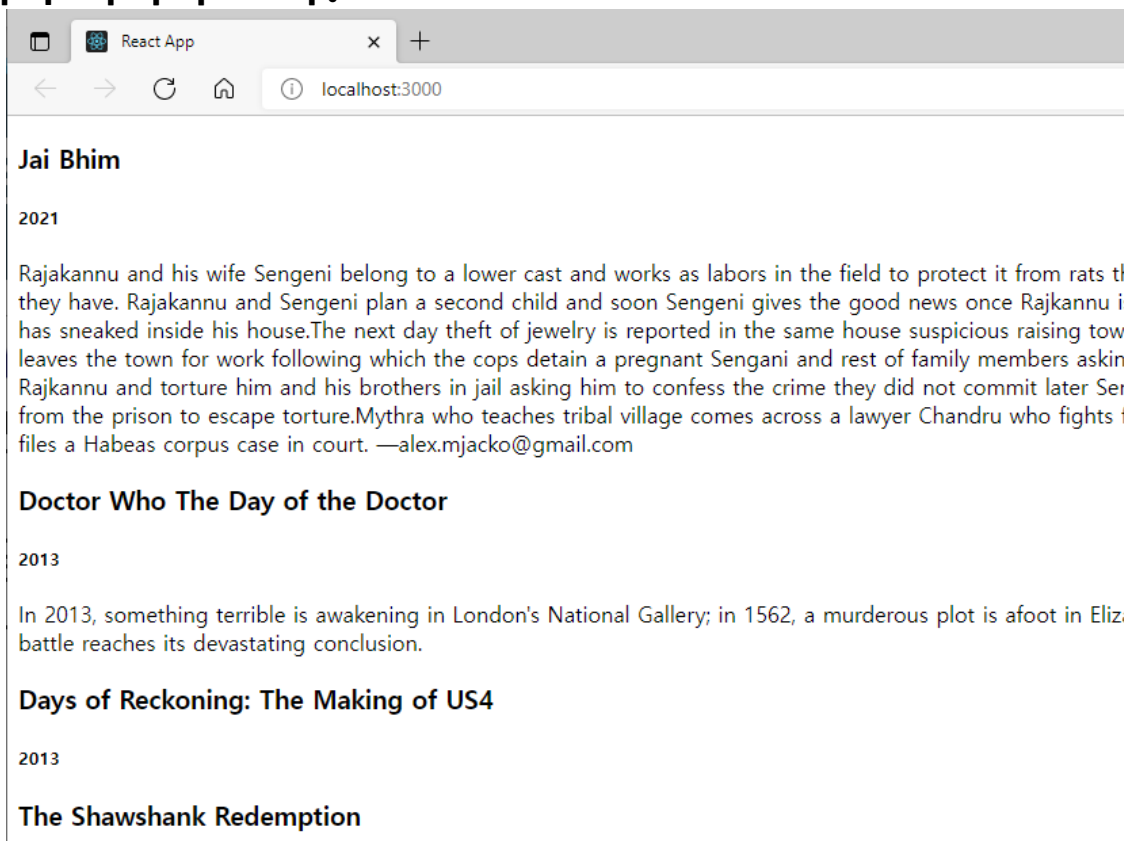
- **Movie** 컴포넌트에도 **HTML**을 추가해 보자.
- **Movie** 컴포넌트가 반환할 **JSX**를 `<div class="movien"></div>`로 감싸고, 그 안에서 **title**, **year**, **summary**를 목적에 맞는 엘리먼트로 감싸자.

14	<b>Movie.propTypes = {</b>
15	<b>id: propTypes.number.isRequired,</b>
16	<b>year: propTypes.number.isRequired,</b>
17	<b>title: propTypes.string.isRequired,</b>
18	<b>summary: propTypes.string.isRequired,</b>
19	<b>poster: propTypes.string.isRequired,</b>
20	<b>};</b>
21	
22	<b>export default Movie;</b>

# 4 영화 앱 스타일링 하기 - 기초

## ❖ Movie 컴포넌트에 HTML 추가하기

- 영화 앱을 실행해 보면 **title, year, summary** 정보가 다른 스타일로 출력되고 있다.
- 영화 앱의 형태가 조금씩 만들어지는 느낌이 드는가?
- 계속해서 나아가 보자.

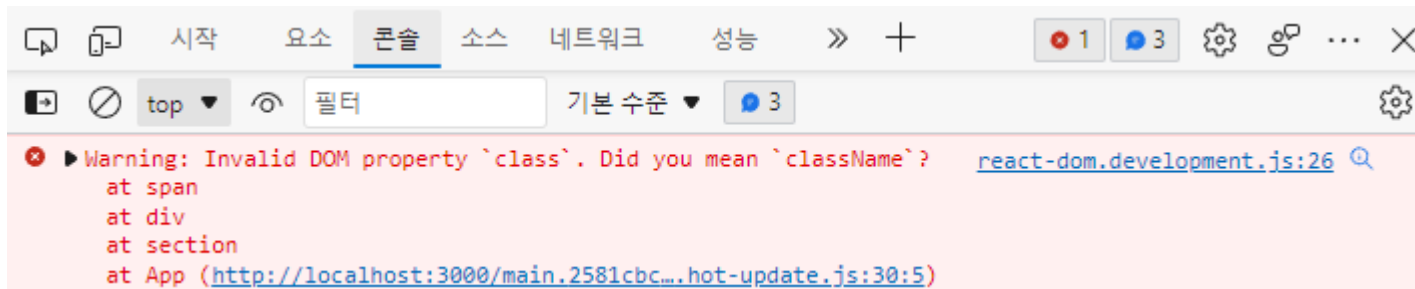




## 4 영화 앱 스타일링 하기 - 기초

### ❖ class 속성 관련 경고 메시지

- 여기까지 영화 앱을 만들고 혹시 [콘솔] 탭을 열었다면 class 속성 관련 경고 메시지를 보았을 것이다.



- 이 메시지는 JSX에서 반환할 엘리먼트에 'class'라는 이름으로 class 속성을 사용했기 때문에 나타난 것이다.
- HTML에서는 class 속성을 사용하려면 'class'를 그대로 쓰지만, 리액트에서 class 속성을 사용하기 위해서는 'class'가 아니라 'className'을 써야한다.
- 물론 이 경고 메시지도 스타일링을 마친 뒤 고칠 예정이니 너무 걱정 말자.

## 4 영화 앱 스타일링 하기 - 기초

### ❖ 영화 포스터 이미지 추가하기

- 아직 영화 앱에 포스터 이미지가 추가되지 않았다.
- 이제 `poster props`를 추가해 보자.
- 전체 엘리먼트를 감싸는 `div` 엘리먼트(`class`가 `movie`인)를 추가하고 `img` 엘리먼트를 `<div class="movie__data">` 위에 추가해서 `img` 엘리먼트의 `src` 속성에는 `poster props`를, `alt`, `title` 속성에는 `title props`를 전달했다.

	...
4	function Movie(id, title, year, summary, poster){
5	return(
6	<div class="movie">
7	<img src={poster} alt={title} title={title} />
8	<div class="movie__data">
9	<h3 class="movie__title">{title}</h3>
10	<h5 class="movie__year">{year}</h5>
11	<p class="movie__summary">{summary}</p>
12	</div>
13	</div>
14	);

## 4 영화 앱 스타일링 하기 - 기초

### ❖ 영화 포스터 이미지 추가하기

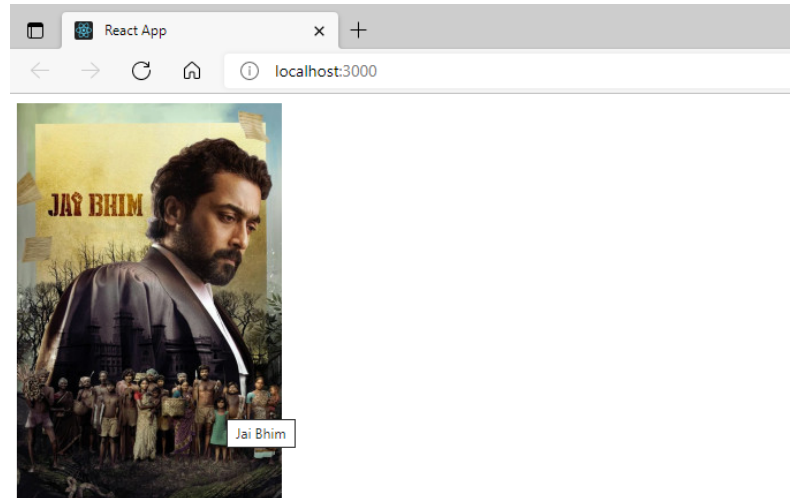
- 아직 영화 앱에 포스터 이미지가 추가되지 않았다.
- 이제 **poster props**를 추가해 보자.
- 전체 엘리먼트를 감싸는 **div** 엘리먼트(class가 **movie**인)를 추가하고 **img** 엘리먼트를 `<div class="movie__data">` 위에 추가해서 **img** 엘리먼트의 **src** 속성에는 **poster props**를, **alt**, **title** 속성에는 **title props**를 전달했다.

```
15 }  
16  
17 Movie.propTypes = {  
18   id: propTypes.number.isRequired,  
19   year: propTypes.number.isRequired,  
20   title: propTypes.string.isRequired,  
21   summary: propTypes.string.isRequired,  
22   poster: propTypes.string.isRequired,  
23 };  
24  
25 export default Movie;
```

## 4 영화 앱 스타일링 하기 - 기초

### ❖ 영화 포스터 이미지 추가하기

- 이제 영화 포스터 이미지까지 나온다.
- 아까 `img` 엘리먼트에 `alt`, `title` 속성을 추가했었다.
- 그 값은 마우스 커서를 이미지 위에 올리면 확인할 수 있다.
- 이제 `Movie` 컴포넌트에 필요한 `HTML`을 모두 작성했다.



**Jai Bhim**

2021

Rajakannu and his wife Sengeni belong to a lower cast and works as labors in the field to provide for their family. They have a son, Rajakannu, and plan a second child. Soon Sengeni gives birth to a healthy baby. However, a thief has sneaked inside their house. The next day, theft of jewelry is reported in the same house. Suspecting the thief, Rajakannu leaves the town for work. Following this, the cops detain a pregnant Sengeni and the rest of the family. Rajakannu is tortured and his brothers are in jail, asking him to confess the crime they did not commit. From the prison, he escapes torture. Mythra, who teaches in a tribal village, comes across a lawyer, Chakrabarti, who files a Habeas corpus case in court. —alex.mjacko@gmail.com

## 4 영화 앱 스타일링 하기 - 기초

### ❖ Movie 컴포넌트 정리하기

- 그런데 코드를 완성하니 **Movie** 컴포넌트에는 **id props**가 필요하지 않다.
- 개발하다 보면 이런 일이 자주 있을 것이다.
- **id props**를 제거해주자.

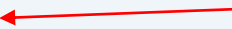
여기를 지우면 된다.

	...
4	function Movie( title, year, summary, poster){
5	return(
6	<div class="movie">
7	<img src={poster} alt={title} title={title} />
8	<div class="movie__data">
9	<h3 class="movie__title">{title}</h3>
10	<h5 class="movie__year">{year}</h5>
11	<p class="movie__summary">{summary}</p>
12	</div>
13	</div>
14	);
15	}
16	

## 4 영화 앱 스타일링 하기 - 기초

### ❖ Movie 컴포넌트 정리하기

- 그런데 코드를 완성하니 **Movie** 컴포넌트에는 **id props**가 필요하지 않다.
- 개발하다 보면 이런 일이 자주 있을 것이다.
- **id props**를 제거해주자.

17	...
18	Movie.propTypes = {
19	 여기도 지워야 한다.
20	year: propTypes.number.isRequired,
21	title: propTypes.string.isRequired,
22	summary: propTypes.string.isRequired,
23	poster: propTypes.string.isRequired,
24	};
25	
26	export default Movie;

## 4 영화 앱 스타일링 하기 - 기초

### ❖ Movie 컴포넌트 정리하기

- **Movie** 컴포넌트를 정리했다면 이제 **Movie** 컴포넌트를 예쁘게 만들어 볼 차례다.
- 리액트에서는 **CSS**를 적용할 수 있는 방법이 많이 있는데 여기서는 가장 기초적인 방법으로 **CSS**를 적용할 것이다.

## 4 영화 앱 스타일링 하기 - 기초

### ❖ style 속성으로 title 스타일링하기

- title props를 포함하고 있는 h3 엘리먼트에 style 속성을 추가하고 {{backgroundColor : "red"}}와 같이 속성값을 입력하자.

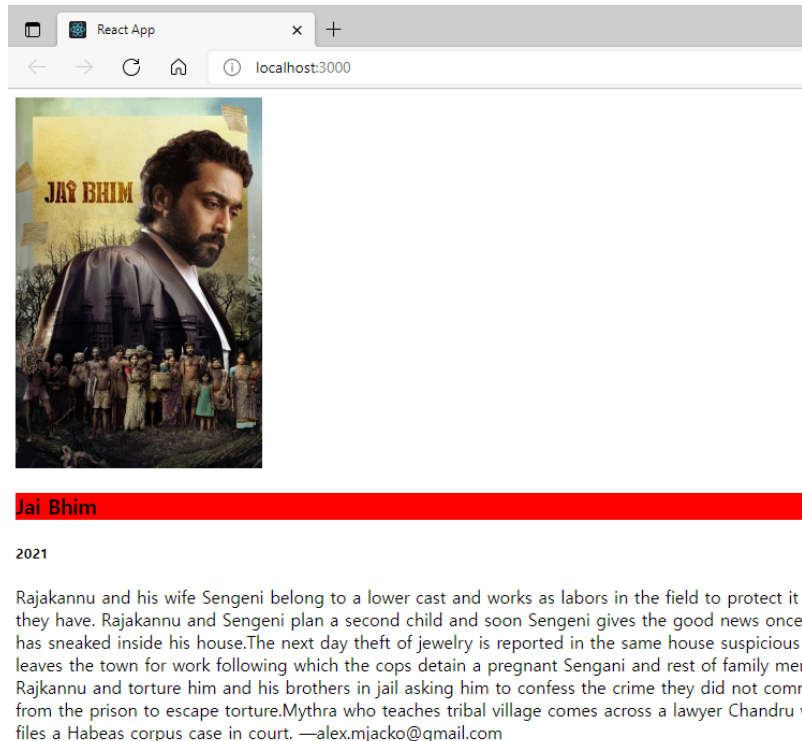
	...
4	function Movie( title, year, summary, poster){
5	return(
6	<div class="movie">
7	<img src={poster} alt={title} title={title} />
8	<div class="movie__data">
9	<h3 class="movie__title" style={{backgroundColor: 'red'}}>
10	{title}
11	</h3>
12	<h5 class="movie__year">{year}</h5>
13	<p class="movie__summary">{summary}</p>
14	</div>
15	</div>
16	);



# 4 영화 앱 스타일링 하기 - 기초

## ❖ style 속성으로 title 스타일링하기

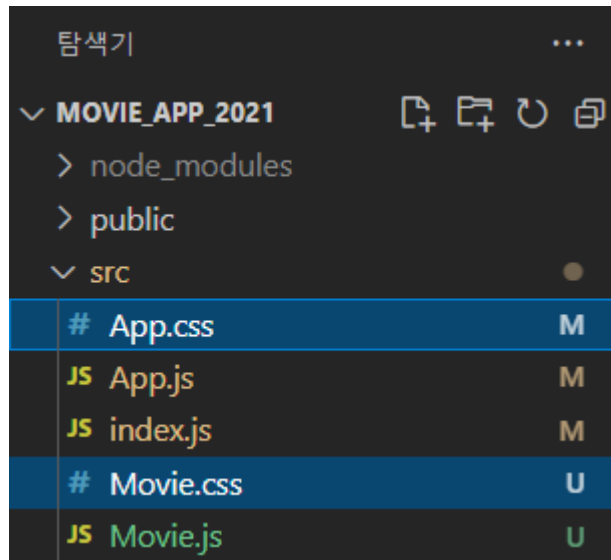
- title에 빨간색 배경이 추가되었다.
- 이걸 style 속성을 이용한 것이다.
- 하지만 HTML과 CSS에 익숙하다면 CSS 파일을 만드는 방식에 익숙할 것이다.
- CSS 파일을 생성하고 임포트하는 방법을 알아보자.



## 4 영화 앱 스타일링 하기 - 기초

### ❖ CSS 파일 생성하기

- src 폴더에 **Movie.css**, **App.css** 파일을 만들자.



## 4 영화 앱 스타일링 하기 - 기초

### ❖ App, Movie 컴포넌트에 CSS 파일 импорт하기

- App, Movie 컴포넌트에 App.css, Movie.css를 각각 импорт하자.
- 그냥 맨 위에 1줄씩만 추가하면 된다.
- Movie 컴포넌트에 적용한 style 속성은 사용하지 않을 거니까 지우자.
- App.js

1	import React from "react";
2	import axios from "axios";
3	import Movie from "./Movie";
4	import './App.css';
5	
6	class App extends React.Component{
	...
50	}
51	
52	export default App;

## 4 영화 앱 스타일링 하기 - 기초

### ❖ App, Movie 컴포넌트에 CSS 파일 импорт하기

- App, Movie 컴포넌트에 App.css, Movie.css를 각각 импорт하자.
- 그냥 맨 위에 1줄씩만 추가하면 된다.
- Movie 컴포넌트에 적용한 style 속성은 사용하지 않을 거니까 지우자.
- Movie.js

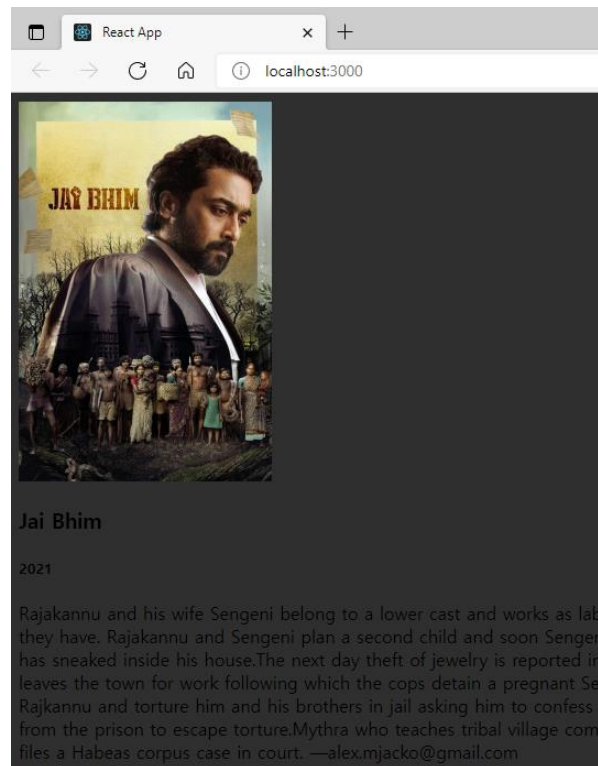
1	import React from "react";
2	import PropTypes from "prop-types";
3	import './Movie.css';
4	
5	function Movie({ title, year, summary, poster }) {
6	return (
7	<div class="movie">
8	<img src={poster} alt={title} title={title} />
9	<div class="movie__data">
10	<h3 class="movie__title">
11	{title}
12	</h3>
13	<h5 class="movie__year">{year}</h5>
14	...

# 4 영화 앱 스타일링 하기 - 기초

## ❖ App.css 파일 작성하기

- App.css 파일을 수정해보자.
- 배경색을 어둡게 바꿔보자.

1	<code>body{</code>
2	<code>background-color: #2f2f2f;</code>
3	<code>}</code>





**Thank You !**