

삼성청년 SW·AI아카데미

JavaScript

<알림>

본 강의는 삼성청년SW·AI아카데미의 콘텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사, 촬영,
녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

Day4-1. Ajax

챕터의 포인트

- Promise
- Ajax 개요
- fetch
- axios
- async / await

Promise

Async 함수들로 Sync 동작을 구현하는 코드 = Callback Hell

- 이해하기 힘들 정도로 가독성이 떨어져, 유지보수가 힘들다.

```
1 function hell(win) {  
2   // for listener purpose  
3   return function() {  
4     loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {  
5       loadLink(win, REMOTE_SRC+'/lib/async.js', function() {  
6         loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {  
7           loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {  
8             loadLink(win, REMOTE_SRC+'/lib/underscore.min.js', function() {  
9               loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {  
10                loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {  
11                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {  
12                  loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {  
13                   async.eachSeries(SRIPTS, function(src, callback) {  
14                     loadScript(win, BASE_URL+src, callback);  
15                   });  
16                 });  
17               });  
18             });  
19           });  
20         });  
21       });  
22     });  
23   });  
24 });  
25 };  
26 }
```



출처 : <https://medium.com/@quyetvv/async-flow-from-callback-hell-to-promise-to-async-await-2da3ecff997>

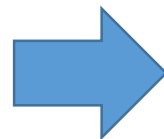
Promise

- 비동기 작업을 좀 더 편하게 할 수 있도록 ES6에서 도입
- 콜백 방식의 콜백지옥을 보완
- `new Promise((resolve, reject) => {});`
- **pending 상태**
 - 대기 상태, resolve나 reject가 실행되기 전의 상태
- **fulfilled 상태**
 - 이행 상태, resolve 호출 시 의 상태
- **rejected 상태**
 - 실패 상태, reject 호출 시의 상태



setTimeout을 Promise로 만들기

```
function promiseSetTimeout(time) {  
  return new Promise((resolve, reject) => {  
    return setTimeout(() => {  
      resolve(1);  
    }, time);  
  });  
}
```



```
promiseSetTimeout(1000).then(li => {  
  console.log(li);  
})  
► Promise {<pending>}  
1
```


Promise Chaining

- Promise.then을 호출하면 자동적으로 Promise가 return 된다
- Return을 통해 값을 넘겨 받을 수 있다.

```
· promiseSetTimeout(1000).then(li =>{  
    console.log(li);  
    return li +1;  
}).then(list =>{  
    console.log(list);  
    return list+1;  
})
```

```
· ▶ Promise {<pending>}
```

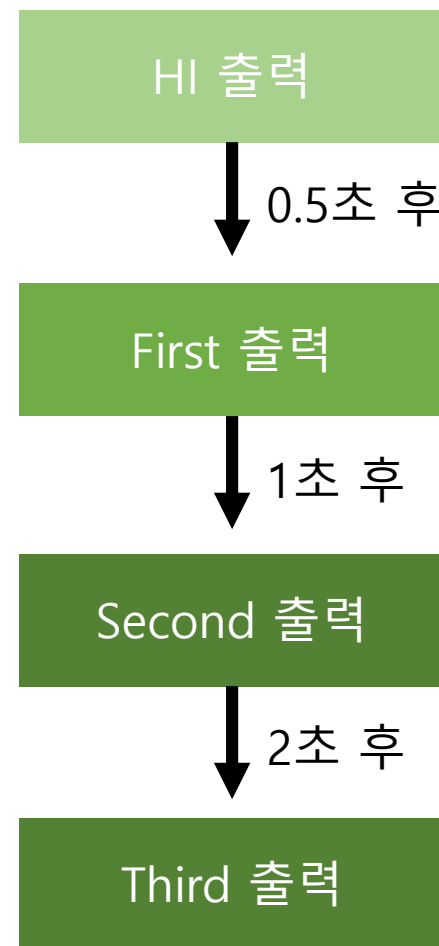
1

2

setTimeout 을 사용하여 Sync(동기) 동작 구현

1. 웹페이지가 로딩 되자마자 console.log 에 HI 출력
2. 0.5 초 후 First가 콘솔에 출력
3. 1 초 후 Second가 콘솔에 출력
4. 2 초 후 Third가 콘솔에 출력

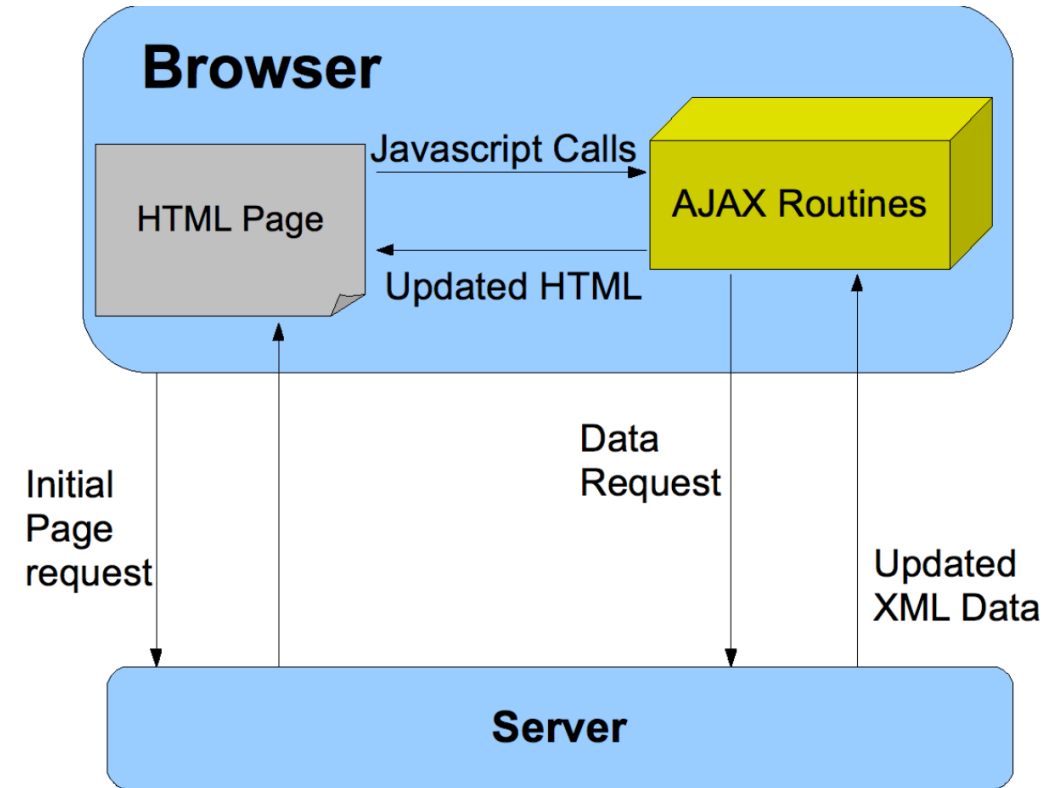
▶	⊘	top
HI		
First		
Second		
Third		
>		



Ajax 개요

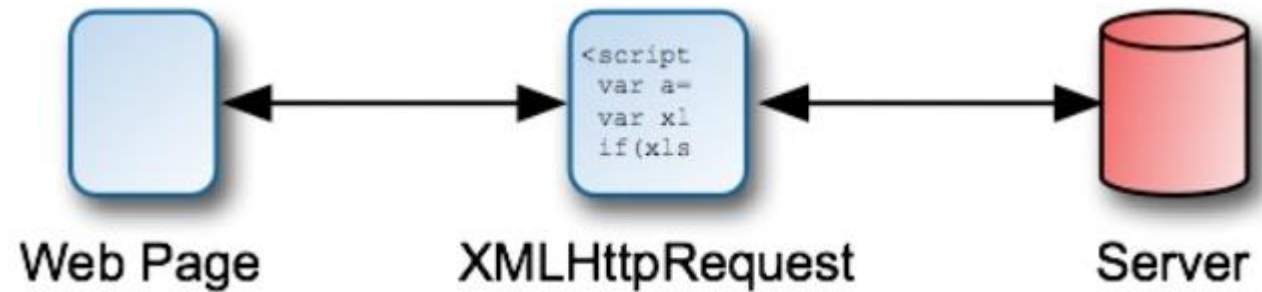
Ajax(Asynchronous JavaScript and XML)

- 서버와 통신하기 위해 XMLHttpRequest 객체 사용
- 비동기성으로 페이지 새로고침을 하지 않고도 수행된다.



XMLHttpRequest 를 활용해 Ajax 통신해보기

- XMLHttpRequest를 통해 비동기 Ajax통신의 기초를 익힌다



fetch

fetch API

- XMLHttpRequest와 유사하나 더 발전된 API (es6 도입)
- 객체를 Promise 형태로 반환 받는다.
- json의 타입별로 쉽게 적용이 가능하다.

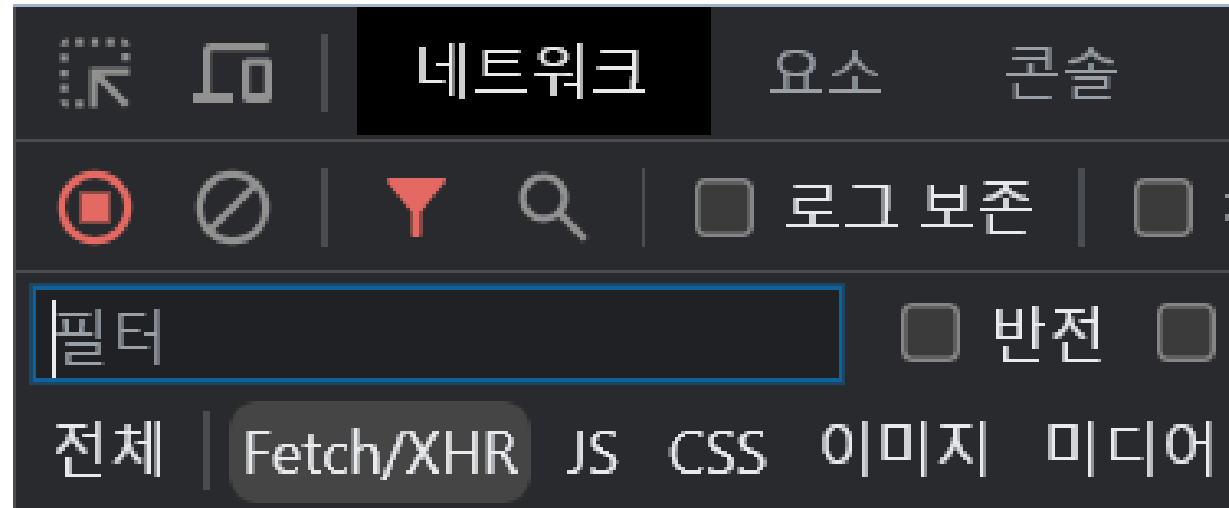
```
const url = "https://jsonplaceholder.typicode.com/todos/";

fetch(url)
  .then((res) => res.json())
  .then((res) => console.log(res))
  .catch((err) => console.log(err));
```

개발자 도구 네트워크 탭

- Fetch/XHR

- Fetch 함수와 XMLHttpRequest 의 줄임말로 비동기 통신만 조회 가능



axios

가장 널리 쓰이는 http 통신 라이브러리

- Vue/React 에서도 권고 라이브러리로 axios가 지정이 되어 있다.
- Promise 형태 리턴
- 사용하기 편리하다.

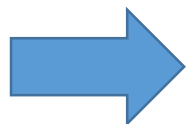
```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
</head>
<body>

  <script>
    const url = "https://jsonplaceholder.typicode.com/todos/";
    axios.get(url).then((res) => {
      console.log(res);
    });
```

Axios 활용하기

- <https://jsonplaceholder.typicode.com/users/3> 호출
 - users/3 의 id값을 받아온다
 - 받아온 id값을 <https://jsonplaceholder.typicode.com/posts?userId=id> 에 요청
 - Ex) <https://jsonplaceholder.typicode.com/posts?userId=3>

```
{
  "id": 3,
  "name": "Clementine Bauch",
  "username": "Samantha",
  "email": "Nathan@yesenia.net",
  "address": {
    "street": "Douglas Extension",
    "suite": "Suite 847",
    "city": "McKenziehaven",
    "zipcode": "59590-4157",
    "geo": {
      "lat": "-68.6102",
      "lng": "-47.0653"
    }
  },
  "phone": "1-463-123-4447",
  "website": "ramiro.info",
  "company": {
    "name": "Romaguera-Jacobson",
    "catchPhrase": "Face to face bifurcated interface",
    "bs": "e-enable strategic applications"
  }
}
```



```
{
  "userId": 3,
  "id": 21,
  "title": "asperiores ea ipsam voluptatibus modi minima quia sint",
  "body": "repellat aliquid praesentium dolorem quo#nsed totam minus non itaque#nnihil labore molestiae sunt dolor eveniet hic recusandae veniam#ntempora et tenetur expedita sunt"
},
{
  "userId": 3,
  "id": 22,
  "title": "dolor sint quo a velit explicabo quia nam",
  "body": "eos qui et ipsum ipsam suscipit aut#nsed omnis non odio#nexedita earum mollitia molestiae aut atque rem suscipit#nnam impedit esse"
},
{
  "userId": 3,
  "id": 23,
  "title": "maxime id vitae nihil numquam",
  "body": "veritatis unde neque eligendi#nquae quod architecto quo neque vitae#nest illo sit tempora doloremque fugit quod#net et vel beatae sequi ullam sed tenetur perspiciatis"
},
{
  "userId": 3,
  "id": 24,
  "title": "autem hic labore sunt dolores incidunt",
  "body": "enim et ex nulla#nomnis voluptas quia qui#nvolutatem consequatur numquam aliquam sunt#ntotam recusandae id dignissimos aut sed asperiores deserunt"
}
```

async / await

async / await

- Promise를 쉽게 사용하기 위해 ES8에 도입
- async는 함수 앞에 위치.
- await는 async 함수 내부에 위치.
- promise를 반환한다
- 비동기 코드를 동기적으로 작성한다

```
const url = "https://jsonplaceholder.typicode.com/todos/";

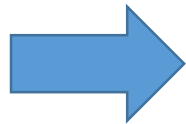
async function start() {
  try {
    const result = await axios.get(url);
    console.log(result.data);
  } catch (error) {
    console.log(error);
  }
}

start();
```

axios + async await

- <https://jsonplaceholder.typicode.com/users/3> 호출
 - users/3 의 id값을 posts/userId=id 에 요청
 - Ex) <https://jsonplaceholder.typicode.com/posts?userId=3>

```
{
  "id": 3,
  "name": "Clementine Bauch",
  "username": "Samantha",
  "email": "Nathan@yesenia.net",
  "address": {
    "street": "Douglas Extension",
    "suite": "Suite 847",
    "city": "McKenziehaven",
    "zipcode": "59590-4157",
    "geo": {
      "lat": "-68.6102",
      "lng": "-47.0653"
    }
  },
  "phone": "1-463-123-4447",
  "website": "ramiro.info",
  "company": {
    "name": "Romaguera-Jacobson",
    "catchPhrase": "Face to face bifurcated interface",
    "bs": "e-enable strategic applications"
  }
}
```



```
{
  "userId": 3,
  "id": 21,
  "title": "asperiores ea ipsam voluptatibus modi minima quia sint",
  "body": "repellat aliquid praesentium dolorem quo#nsed totam minus non itaque#nnihil labore molestiae sunt dolor eveniet hic recusandae veniam#ntempora et tenetur expedita sunt"
},
{
  "userId": 3,
  "id": 22,
  "title": "dolor sint quo a velit explicabo quia nam",
  "body": "eos qui et ipsum ipsam suscipit aut#nsed omnis non odio#nexedita earum mollitia molestiae aut atque rem suscipit#nnam impedit esse"
},
{
  "userId": 3,
  "id": 23,
  "title": "maxime id vitae nihil numquam",
  "body": "veritatis unde neque eligendi#nquae quod architecto quo neque vitae#nest illo sit tempora doloremque fugit quod#net et vel beatae sequi ullam sed tenetur perspiciatis"
},
{
  "userId": 3,
  "id": 24,
  "title": "autem hic labore sunt dolores incidunt",
  "body": "enim et ex nulla#nomnis voluptas quia qui#nvolutatem consequatur numquam aliquam sunt#ntotam recusandae id dignissimos aut sed asperiores deserunt"
}
```

컴포넌트를 만들어서 외부 API + HTML 결합

- <https://picsum.photos/v2/list?page=1&limit=20> 활용

Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Alejandro Escamilla



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



Author : Paul Jarvis



응용 문제

- <https://fakestoreapi.com/products> 활용
- title + price + image 나타내기
- 처음부터 진행한다.

```
[
  {
    "id": 1,
    "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
    "price": 109.95,
    "description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",
    "category": "men's clothing",
    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg",
    "rating": {
      "rate": 3.9,
      "count": 120
    }
  },
  {
    "id": 2,
    "title": "Mens Casual Premium Slim Fit T-Shirts ",
    "price": 22.3,
    "description": "Slim-fitting style, contrast raglan long sleeve, three-button henley placket, light weight & diehard baseball fans. The Henley style round neckline includes a three-button placket.",
    "category": "men's clothing",
    "image": "https://fakestoreapi.com/img/71-3HjGNDUL._AC_SY879._SX._UX._SY._UY_.jpg",
    "rating": {
      "rate": 4.1,
      "count": 259
    }
  }
],
```


Day4-2. Youtube 프로젝트

챕터의 포인트

- Youtube 프로젝트
- 영화 프로젝트

- 총 두 가지 프로젝트가 제공 됨
 1. Youtube 프로젝트
 2. 영화 프로젝트
- 두 프로젝트 모두, 제공되는 REST API 활용해 구현

Youtube 프로젝트

API 발급

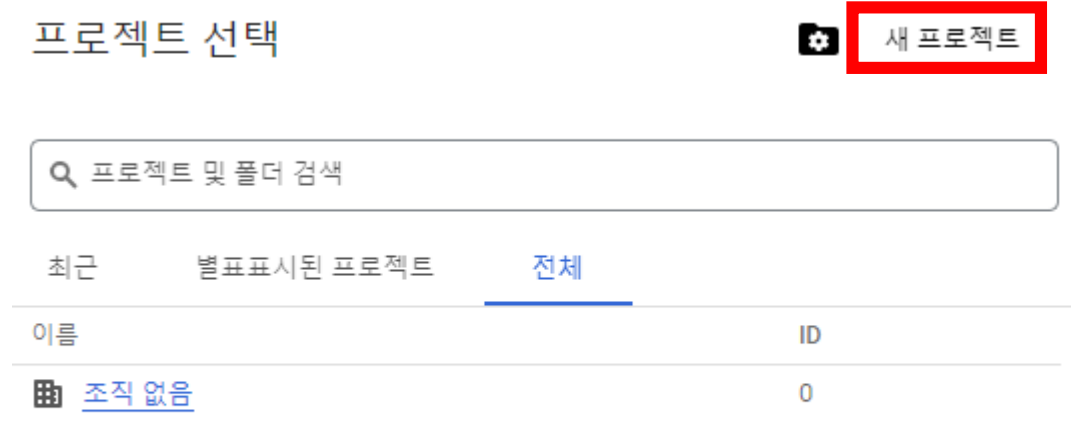
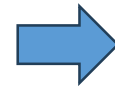
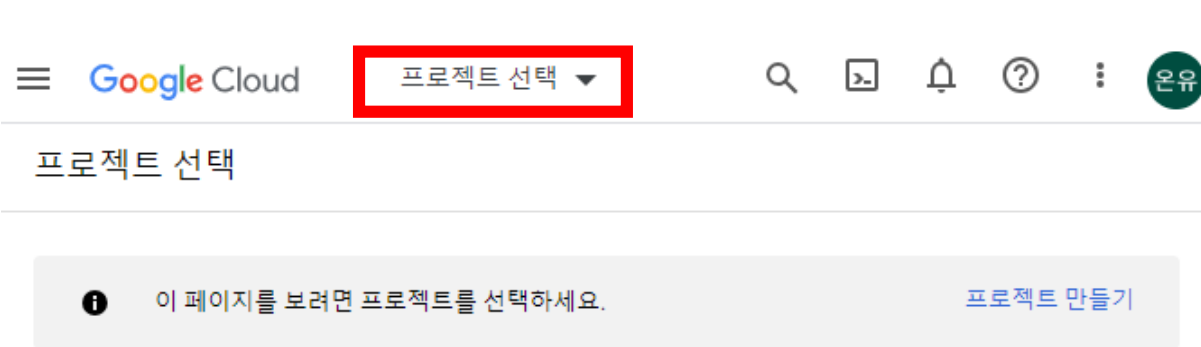
- “Google 클라우드 플랫폼” -> 콘솔 접속

<https://console.cloud.google.com/welcome?authuser=2&project=psychic-rhythm-378805>



API 발급

- "Google 클라우드 플랫폼" -> 콘솔 접속
<https://console.cloud.google.com/welcome?authuser=2&project=psychic-rhythm-378805>
- "Youtube data api v3" 검색
- 프로젝트가 없는 경우 프로젝트 만들기 클릭



API 발급

- 프로젝트가 없는 경우 프로젝트 만들기 클릭


프로젝트 이름 *

My Youtube Project

?

프로젝트 ID: my-youtube-project-395314입니다. 나중에 변경할 수 없습니다. [수정](#)

위치 *

 조직 없음

[찾아보기](#)

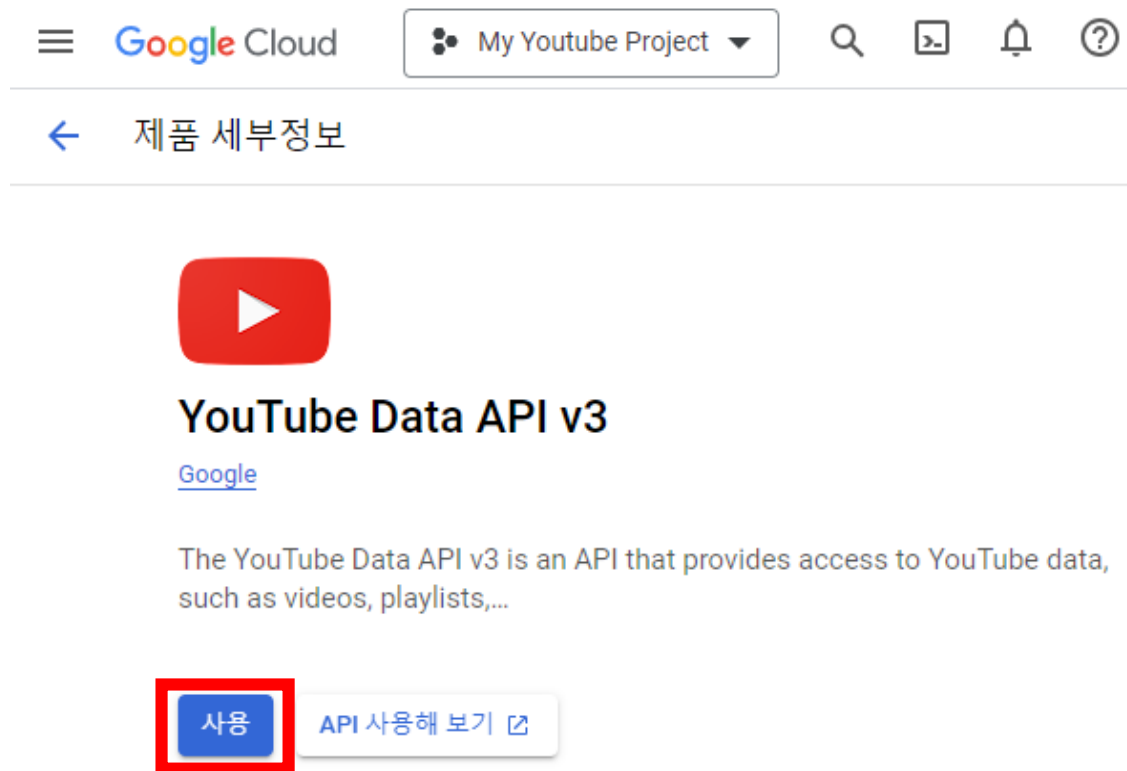
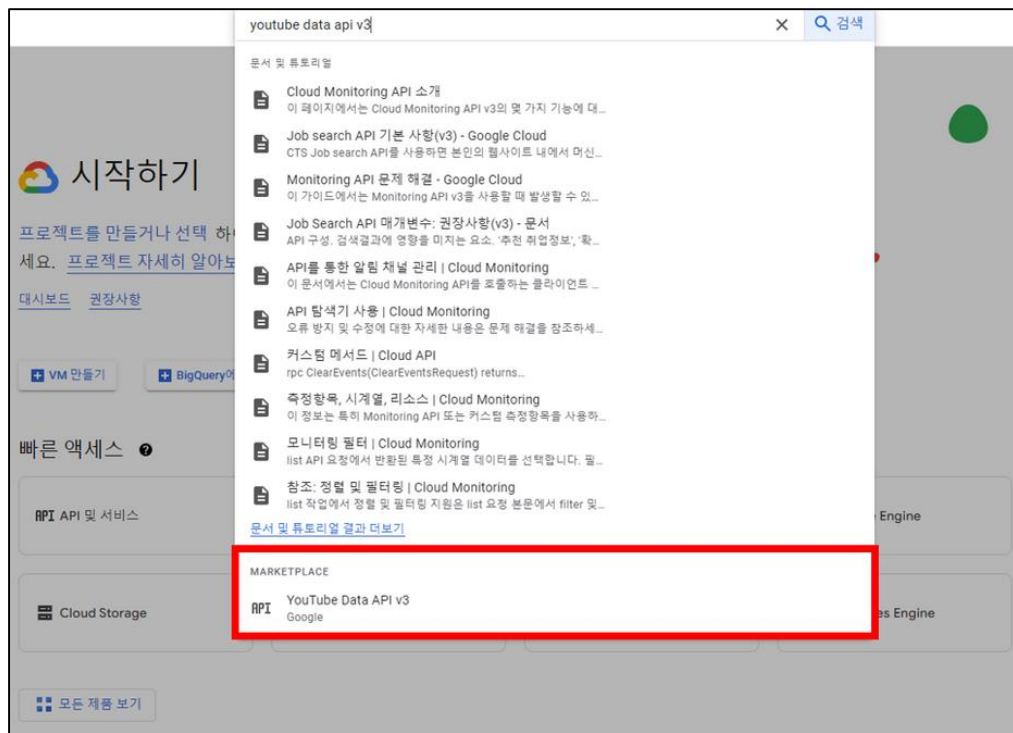
상위 조직 또는 폴더

만들기

취소

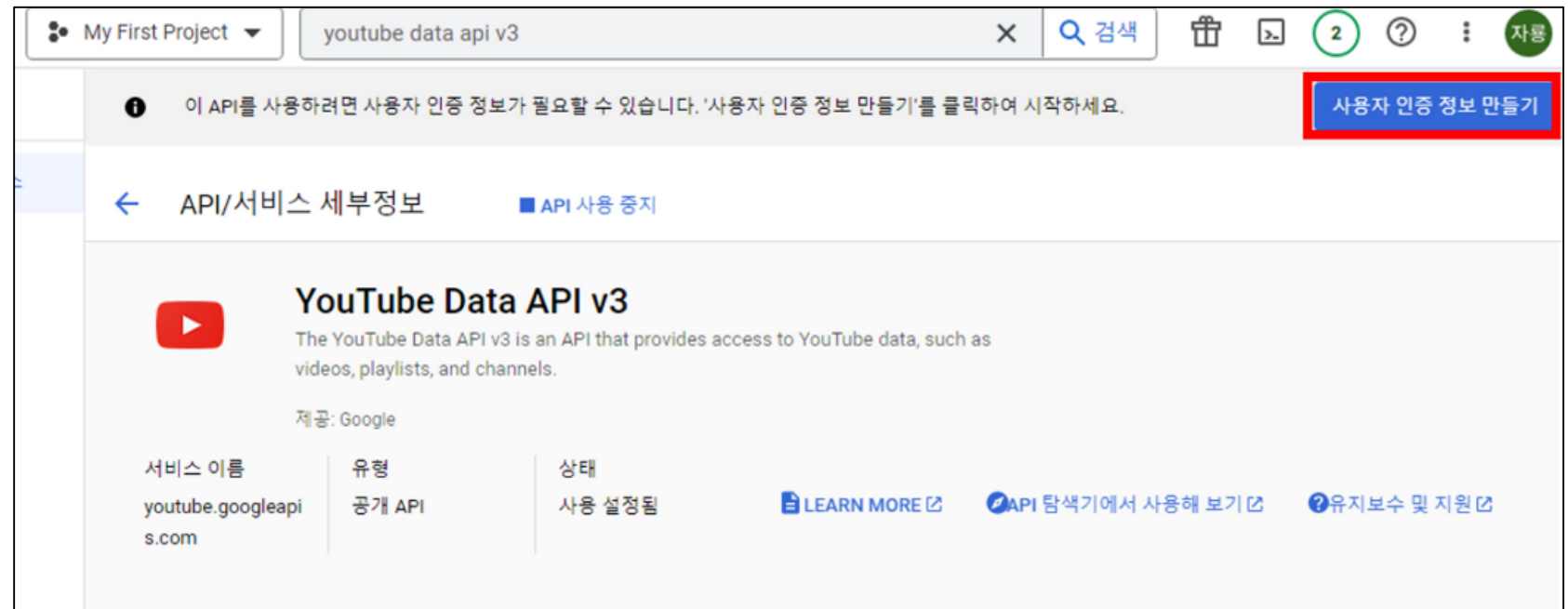
API 발급

- "Youtube data api v3" 검색 - "사용" 버튼 클릭



API 발급

- My First Project 라는 이름의 프로젝트 생성됨
- "사용자 인증 정보 만들기" 버튼 클릭



API 발급

- 공개 데이터로 체크하고, "다음" 버튼 클릭
- API 키 발급 완료 (복사해두기)

사용자 인증 정보 만들기

1 사용자 인증 정보 유형

어떤 API를 사용 중이신가요?

여러 API에서 다른 인증 플랫폼을 사용하고 있으며 일부 사용자 인증 정보가 특정 API만 호출하도록 제한될 수 있습니다.

API 선택 *
YouTube Data API v3

엑세스할 데이터는 무엇인가요? *

요청한 데이터의 유형에 따라 액세스를 승인하는 데 다른 사용자 인증 정보가 필요합니다. [자세히 알아보기](#)

☐ 사용자 데이터
이메일 주소나 연령과 같이 Google 사용자에게 속하는 데이터이며 사용자 동의가 필요합니다. 이를 통해 OAuth 클라이언트가 생성됩니다.

☒ 공개 데이터
레스토랑 정보를 표시하는 공개 지도 데이터와 같이 공개적으로 제공되는 Google 데이터입니다. 이를 통해 API 키가 생성됩니다.

다음

2 내 사용자 인증 정보

완료 취소

✓ 사용자 인증 정보 유형

2 내 사용자 인증 정보

API 키

프로덕션에서 사용하기 전에 이 키를 제한하는 것이 좋습니다.
[RESTRICT KEY](#)

API 키입니다. [사용자 인증 정보 페이지](#)에서 언제든지 확인할 수 있습니다.

API 키
AIzaSyDUWdPs1KvM_27hmLjHnX2Vbq6a70UcXqk

완료 취소

API 테스트

- Youtube API 공식문서 주소
<https://developers.google.com/youtube/v3?hl=ko>
- 프로젝트에선 두 가지 URL 요청 사용 권장
 - /search : 검색어에 해당하는 동영상 검색 결과를 보냄

params	value 예시	설명
key	"발급받은 API 키"	
part	"snippet"	특별한 이유 없으면 "snippet"
type	"video", "playlist", "channel"	검색결과 종류
q	"ssafy"	검색어
maxResults	10	결과 배열의 크기

API 테스트

- /videos : 유튜브 동영상 ID 에 해당하는 동영상 정보를 보냄

params	value 예시	설명
key	"발급받은 API 키"	
part	"snippet"	특별한 이유 없으면 "snippet"
id	"mKKbO_QTFVY"	유튜브 비디오 아이디

API 테스트

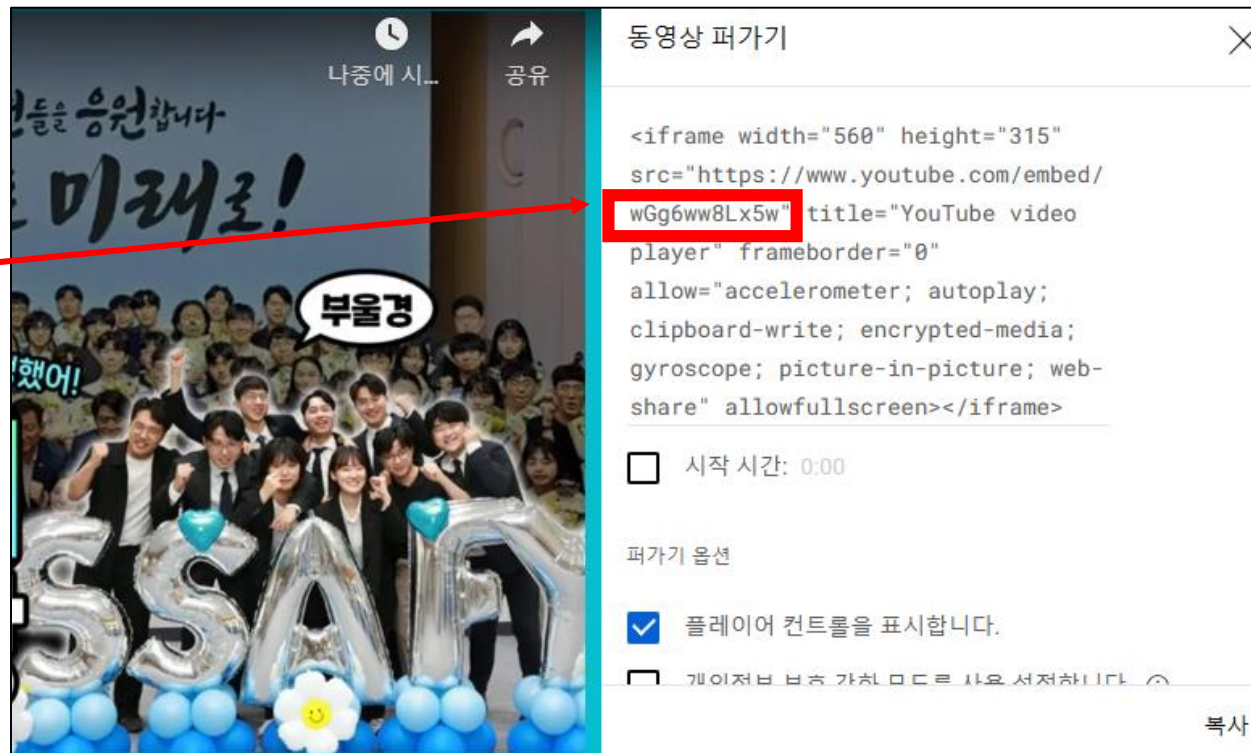
테스트 코드를 입력하고, 결과 확인

```
const url = "https://www.googleapis.com/youtube/v3";
const key = "발급받은API키를입력하세요";
async function getVideos(word) {
  try {
    const response = await axios.get(`${url}/search`, {
      params: {
        key: key,
        part: "snippet",
        type: "video",
        q: word,
        maxResults: 10,
      },
    });
    console.log(response.data.items);
  } catch (error) {
    console.error(error);
  }
}
getVideos("ssafy");
```

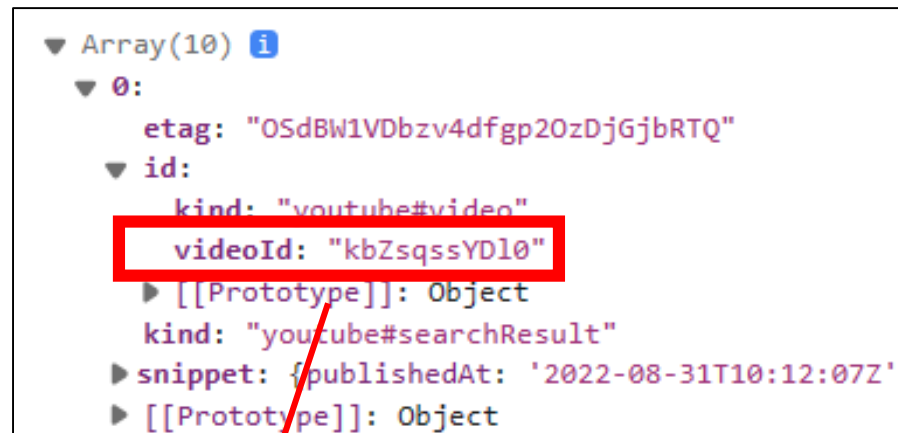
```
▼ (10) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] ⓘ
  ▼ 0:
    etag: "OSdBW1VDbzv4dfgp20zDjGjbRTQ"
    ▶ id: {kind: 'youtube#video', videoId: 'kbZsqssYDl0'}
    kind: "youtube#searchResult"
    ▼ snippet:
      channelId: "UCqPjSHRA0KB4TObEa2k0cRg"
      channelTitle: "재이의날들 daysofjayla"
      description: "안녕하세요, 재이입니다~ 오늘은 평소 예 업로드하던 브이로그와는 다른 영상으로 찾아
      liveBroadcastContent: "none"
      publishTime: "2022-08-31T10:12:07Z"
      publishedAt: "2022-08-31T10:12:07Z"
    ▶ thumbnails: {default: {...}, medium: {...}, high: {...}}
      title: "SSAFY | 싸피 8기가 말해주는 싸피 생활의 모든 것! | 싸밥, 커리큘럼, 수업난이도, 반 분위
    ▶ [[Prototype]]: Object
    ▶ [[Prototype]]: Object
    ▶ 1: {kind: 'youtube#searchResult', etag: 'cvvpHuu2uyaExMBRIC4DDzddaCM', id: {...}, snippet: {...}}
    ▶ 2: {kind: 'youtube#searchResult', etag: 'ubJ8iDm7853XKWR1bDuqMWg3Zow', id: {...}, snippet: {...}}
    ▶ 3: {kind: 'youtube#searchResult', etag: 'D_keIIwcrxYD5R6bhiilFf1nZWo', id: {...}, snippet: {...}}
```

유튜브 videoid 활용

```
▼ Array(10) ⓘ
  ▼ 0:
    etag: "OSdBW1VDbzv4dfgp2OzDjGjbRTQ"
    ▼ id:
      kind: "youtube#video"
      videoId: "kbZsqssYDl0"
      ▶ [[Prototype]]: Object
      kind: "youtube#searchResult"
      ▶ snippet: {publishedAt: '2022-08-31T10:12:07Z',
      ▶ [[Prototype]]: Object
```



유튜브 videoid 활용



```
function getYoutube(id){  
  return `  
    <iframe width="560" height="315" src="https://www.youtube.com/embed/${id.videoId}" title="YouTube video player"  
      frameborder="0"  
      allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share"  
      allowfullscreen>  
    </iframe>  
  `;  
}
```


프로젝트 예시를 참고하여, Youtube API 를 활용한 자유 프로젝트 진행

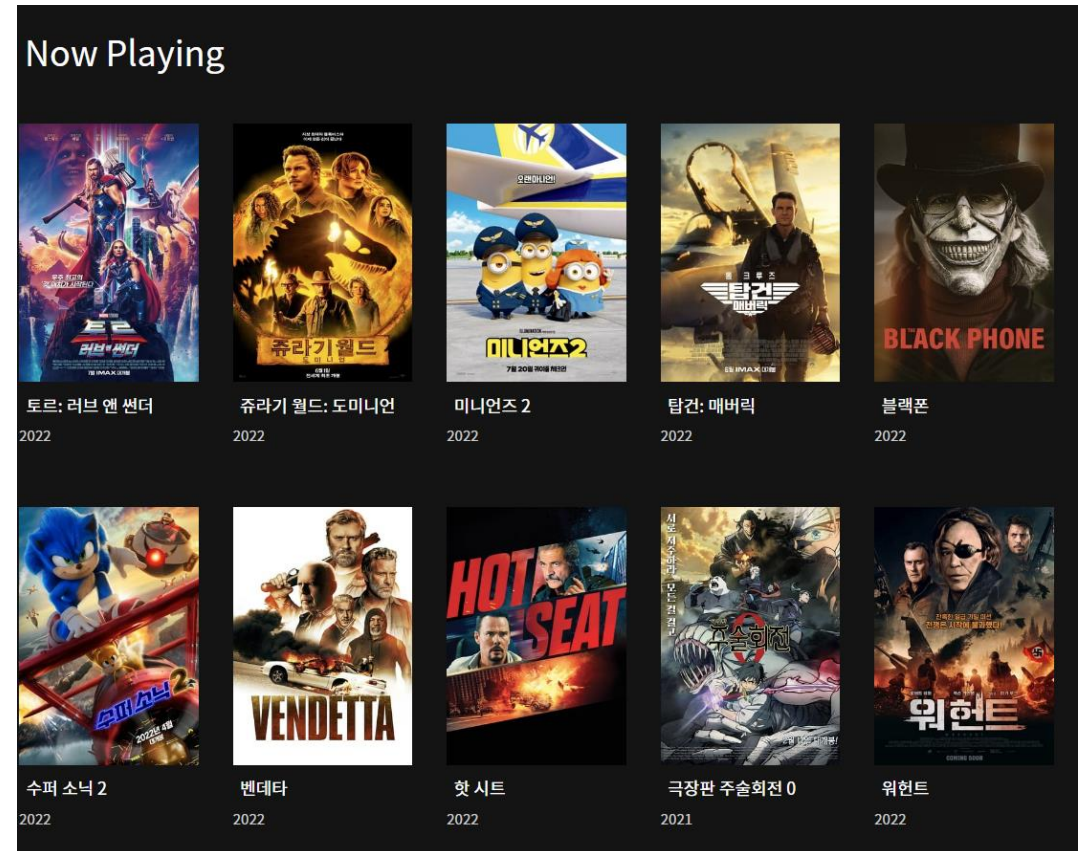


영화 프로젝트

<https://www.themoviedb.org/>

- 가입하기 및 API 발급
- <https://developers.themoviedb.org/3>
- 데이터를 가공해서 본인만의 프로젝트 만들기

```
const request = axios.get("https://api.themoviedb.org/3/movie/now_playing", {  
  params: {  
    api_key: "발급받은 API KEY",  
    language: "ko-KR",  
  }  
})
```



내일 방송에서 만나요!

삼성청년SW·AI아카데미

과제1(10p)

비동기 API로, Sync(동기) 구현하기

과제2(19p)

비동기 중첩 문제 - Axios 활용하기

과제3(22p)

비동기 중첩 문제 - axios + async await

과제4(23p)

컴포넌트를 만들어서 HTML 태그 적용시키기

과제5(24p)

API 호출

과제6(40p)

Youtube API 자유 프로젝트

과제7(42p)

영화 프로젝트 만들기