

220727

# API 통신

할 수 있어

---

HACK YOUR LIFE!

# DB란?

---

디저트배?

## DB란?

---

**Data Base** | 컴퓨터 시스템에 전자적으로 저장되는 구조화된 **정보** 또는 **데이터**의 조직화된 모음

members

memId	role	nickname	name	description
1	호랭이선생님	yuniyuns	윤선영	멋사 10기 프론트 운영진
2	호랭이선생님	jjwzzy	정지원	멋사 10기 프론트 운영진
3	호랭이선생님	soosoo	서지수	멋사 10기 프론트 운영진

## json

```
const data = {  
  members: [  
    {  
      role: "호랭이선생님",  
      memId: 1,  
      nickname: "yuniiyuns",  
      name: "윤선영",  
      description: "멋사 10기 프론트 운영진",  
    },  
    {  
      role: "호랭이선생님",  
      memId: 2,  
      nickname: "jjwzzy",  
      name: "정지원",  
      description: "멋사 10기 프론트 운영진",  
    },  
    {  
      role: "호랭이선생님",  
      memId: 3,  
      nickname: "soosoo",  
      name: "서지수",  
      description: "멋사 10기 프론트 운영진",  
    },  
  ],  
};  
  
export default data;
```

Information.js

```
{  
  "members": [  
    {  
      "role": "호랭이선생님",  
      "memId": 1,  
      "nickname": "yuniiyuns",  
      "name": "윤선영",  
      "description": "멋사 10기 프론트 운영진"  
    },  
    {  
      "role": "호랭이선생님",  
      "memId": 2,  
      "nickname": "jjwzzy",  
      "name": "정지원",  
      "description": "멋사 10기 프론트 운영진"  
    },  
    {  
      "role": "호랭이선생님",  
      "memId": 3,  
      "nickname": "soosoo",  
      "name": "서지수",  
      "description": "멋사 10기 프론트 운영진"  
    },  
  ],  
}
```

Data.json

# 데이터를 어떻게 읽어올까?

---

복습 가보자고~

## map 함수란?

**map 함수** | 반복적인 내용을 효율적으로 보여주고 관리하는  
자바스크립트 배열 객체의 내장 함수  
배열 내 각 요소를 원하는 규칙에 따라 변환한 후  
그 결과로 새로운 배열을 생성한다.  
**=> 데이터를 원하는대로 읽어들인다.**

```
arrays.map((array) => <tag key={array.key}>{array.title}</tag>)
```

↓  
배열이름

↓  
새로운  
배열이름

↓  
태그나  
컴포넌트

↓  
키

↓  
보여줄  
내용

```
const text = ["멋쟁이사자처럼", "중앙대학교", "최고"];

const Box = () => {
  return text.map((t, i) => <StyledBox key={i}>{t}</StyledBox>);
};
```

```
{data.products.map((product) => (
  <StyledBox key={product.id}>
    <img src={product.img} alt="#"></img>
    <text>
      <p>
        <b>{product.title}</b>
      </p>
      <p className="address">{product.address}</p>
      <p className="price">{product.price}</p>
    </text>
  </StyledBox>
)}}}
```

윤선영

정지원

서지수

이서현

유하린

이동길

추교현

권민재

김나예

이의제

조해윤

김정현

권수연

## 실습 가보자고~

**map**함수를 이용해 멤버들의 이름을 불러와서  
버튼 형식으로 만들어보자!



그런데...데이터는 항상 고정적일까...?

습작

습작6

수수 · 방금 전



엉엉ㅠㅠ

내가 새 글을 쓰면...?

blogs

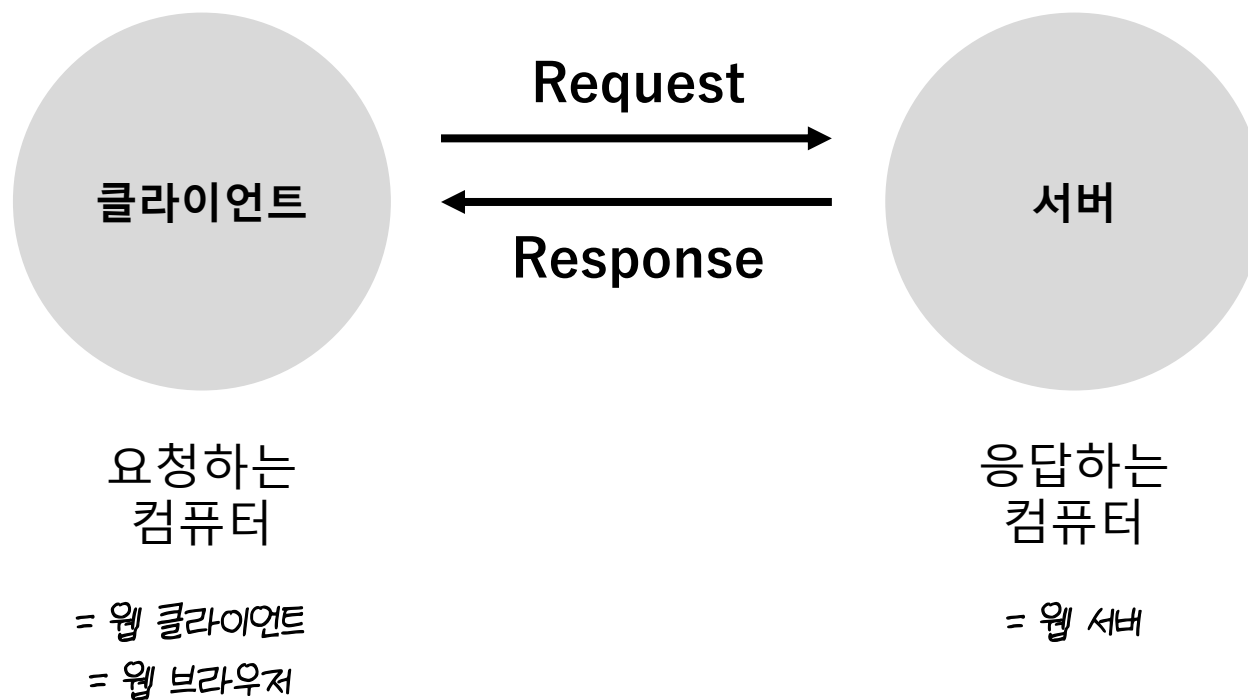
title	postId	nickname	img	description
습작1	1	수수	img/@!\$\$	행복
습작2	2	수수	img/13%\$\$\$%	기뻐
습작3	3	수수	img/23%	좋아
습작4	4	수수	img/!#	최고
습작5	5	수수	img/1@\$#%^	엉엉ㅠㅠ

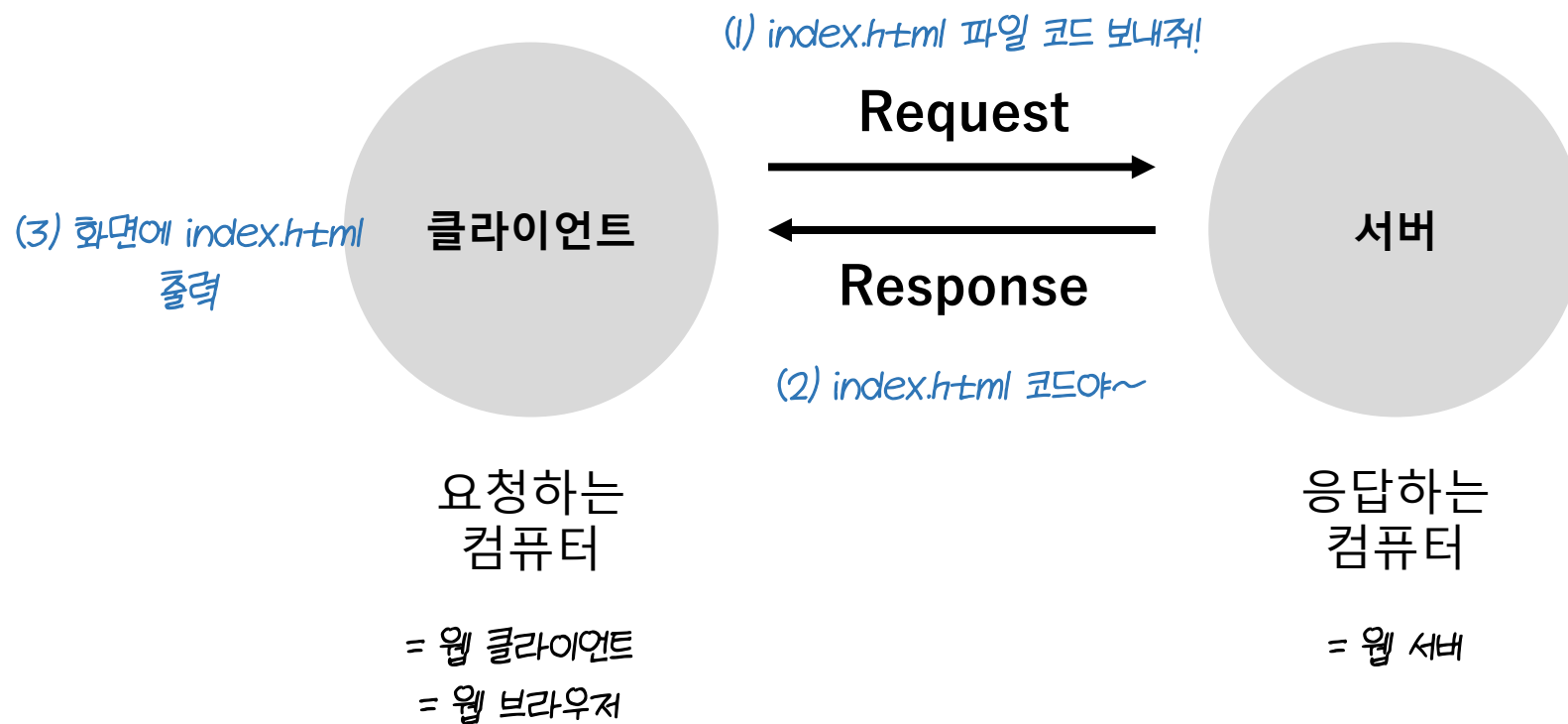
```
{
  "blogs": [
    {
      "title": "습작6",
      "postId": 6,
      "nickname": "수수",
      "img": "img/1@$#%^",
      "description": "엉엉ㅠㅠ"
    }
  ]
}
```

# 서버와 클라이언트

---

개념부터 찹찹 다지고 갑시다

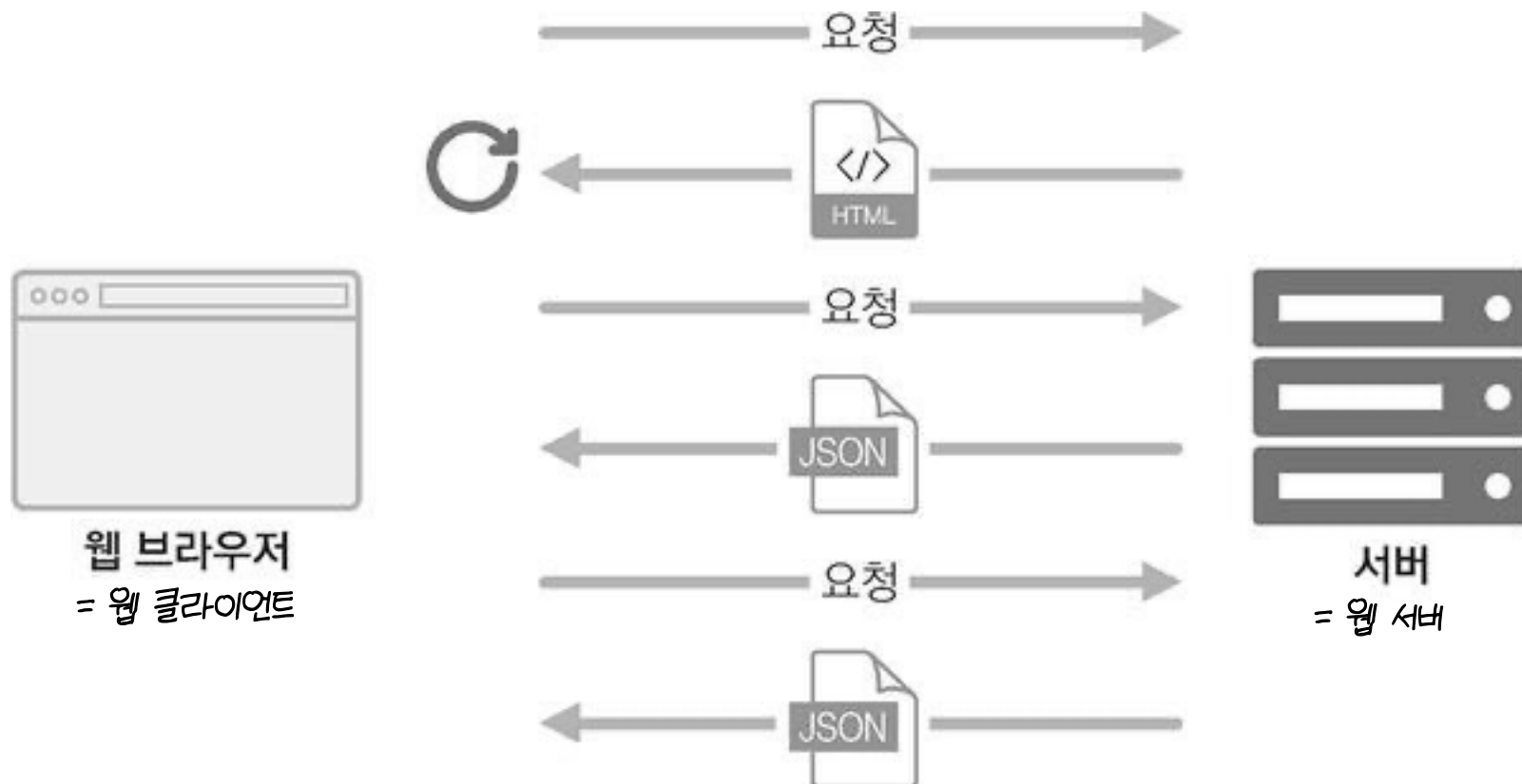


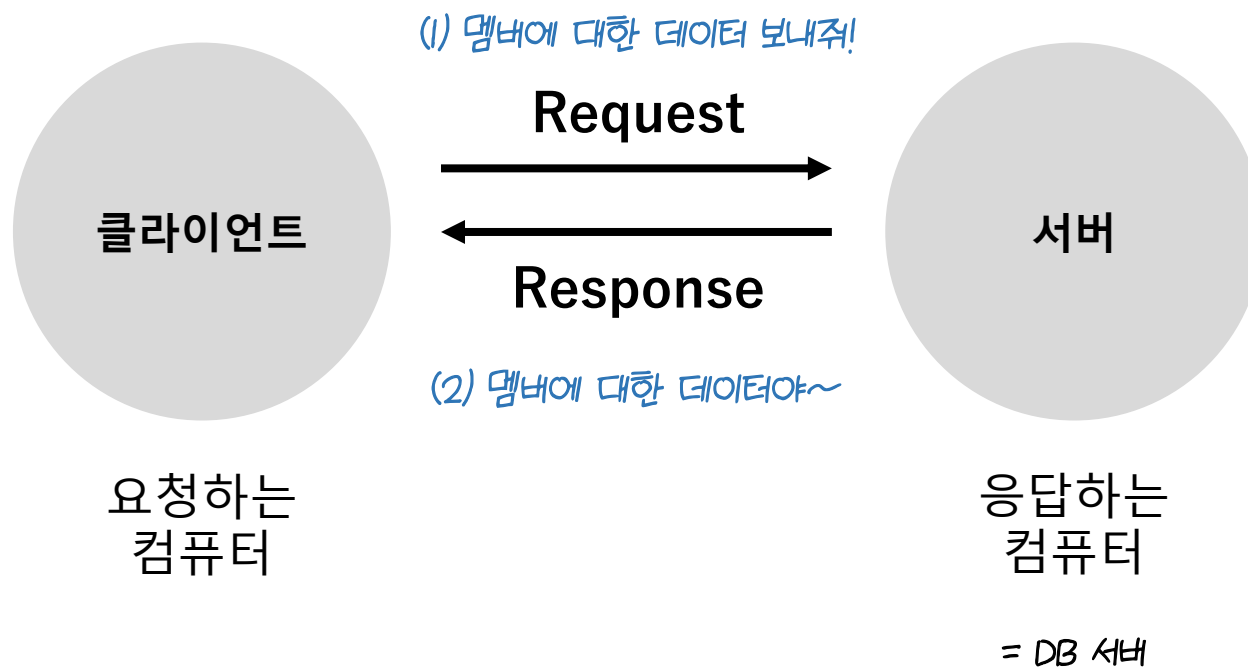


잠깐 복습!

SPA

싱글 페이지 어플리케이션





# API란?

---

## API란?

---

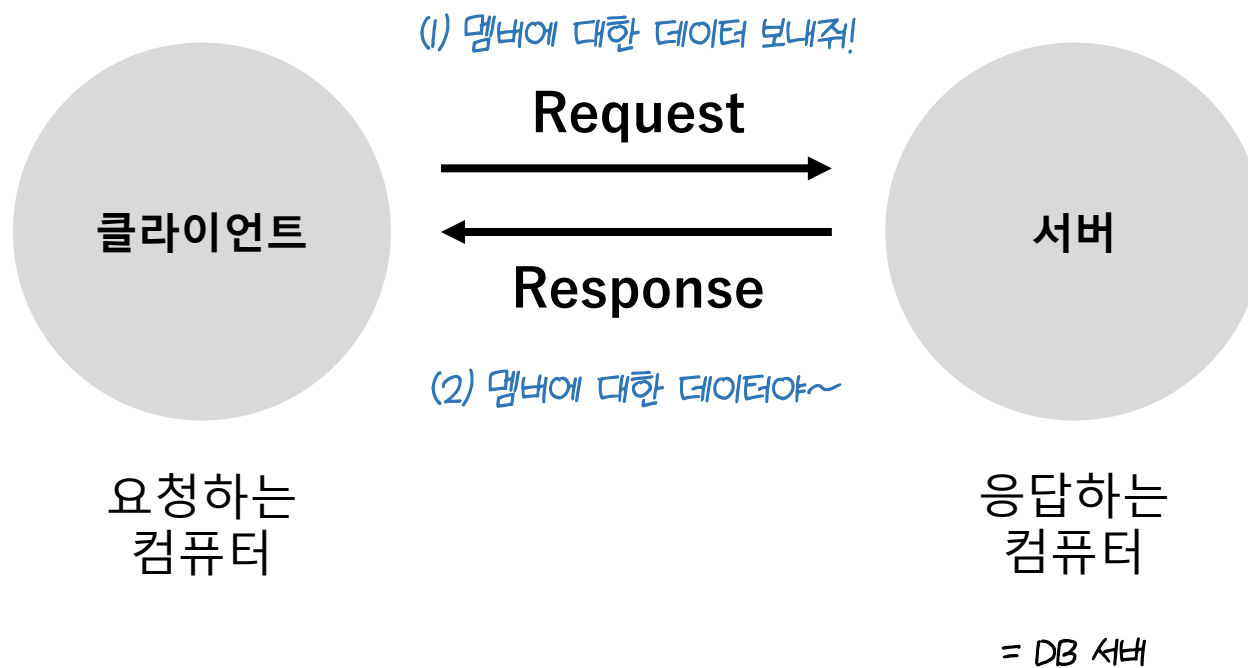
### **Application Programming Interface**

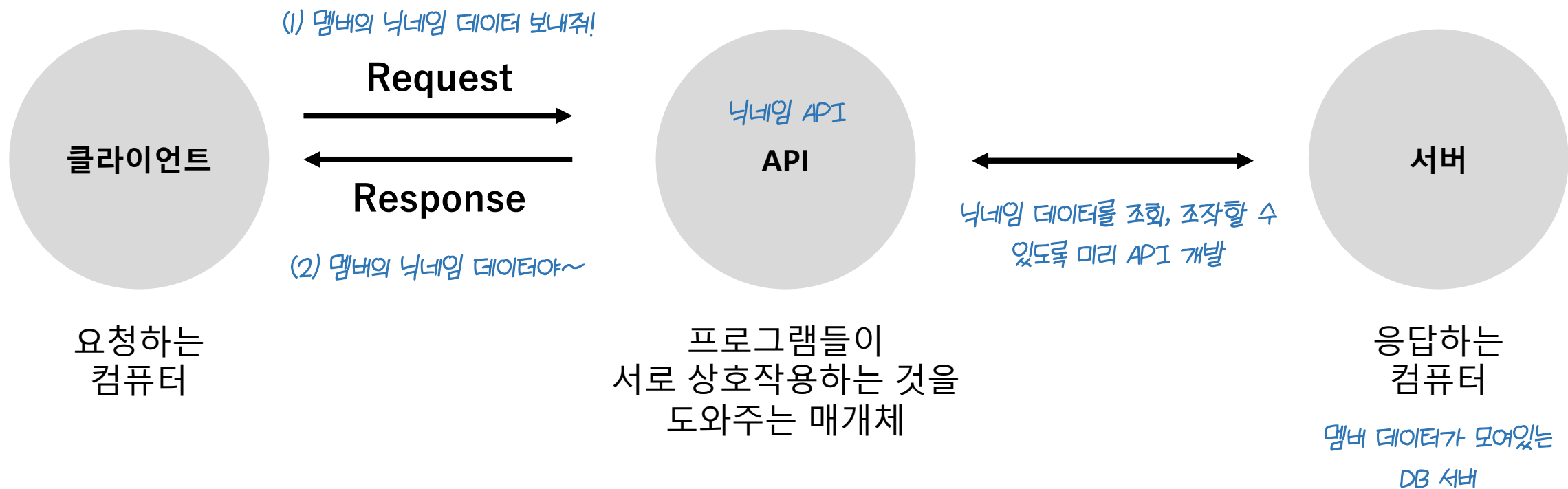
애플리케이션(응용프로그램)에서 사용할 수 있도록,  
운영체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스를 뜻함.  
즉, 애플리케이션이 어떤 프로그램이 제공하는 기능을 사용할 수 있게 만든 매개체  
**컴퓨터나 소프트웨어를 서로 연결함**



- 점원의 역할







## API란?

---

### Application Programming Interface

1. 컴퓨터나 소프트웨어를 서로 **연결**함
2. 출입구 역할
2. 원활하게 통신할 수 있도록

## HTTP API

---

**HTTP** | '웹'이라는 서비스를 이용하기 위해서  
준수해야 되는 **통신 규약**

**HTTP API** | HTTP를 사용하여 프로그램끼리  
소통하는 API

## REST API

---

**REST** | **REST(Representational State Transfer)**란  
자원을 이름으로 구분하여(1 | 데이터의 표현)  
해당 자원의 정보를 주고 받는 (2 | 정보 전달)  
모든 것을 의미

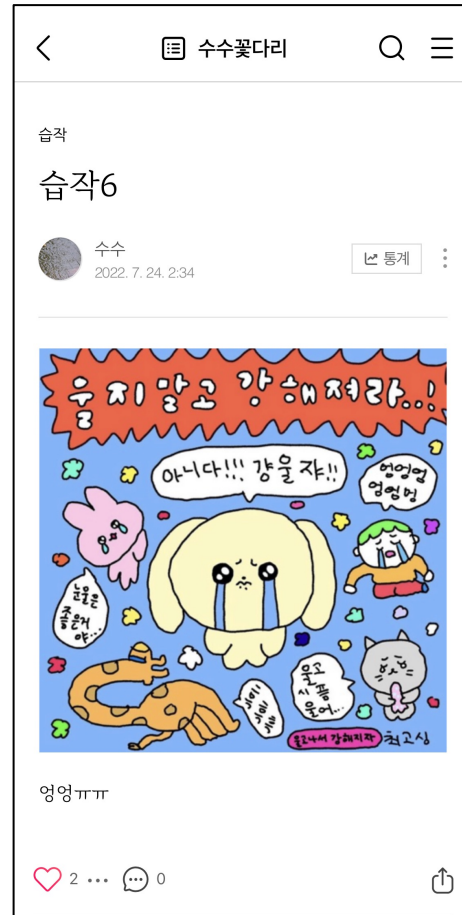
**REST API** | REST는 HTTP를 잘 활용하기 위한  
원칙이라고 할 수 있고  
REST API는 이 원칙을 준수해 만든 API

CRUD | HTTP method

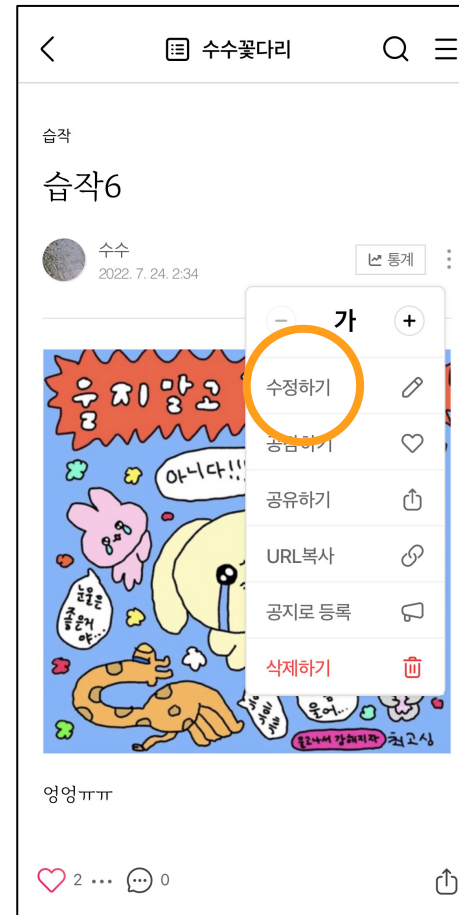
Create		POST
Read		GET
Update		PUT
Delete		DELETE



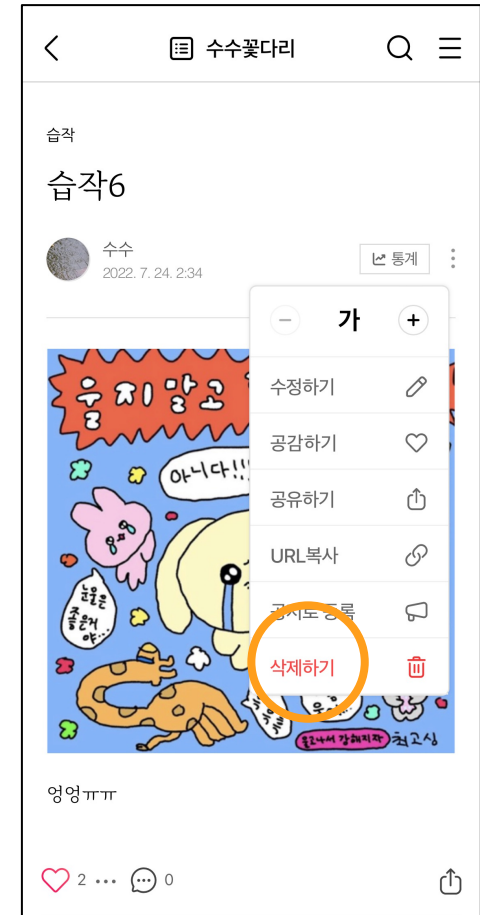
Create | POST



Read | GET



Update | PUT



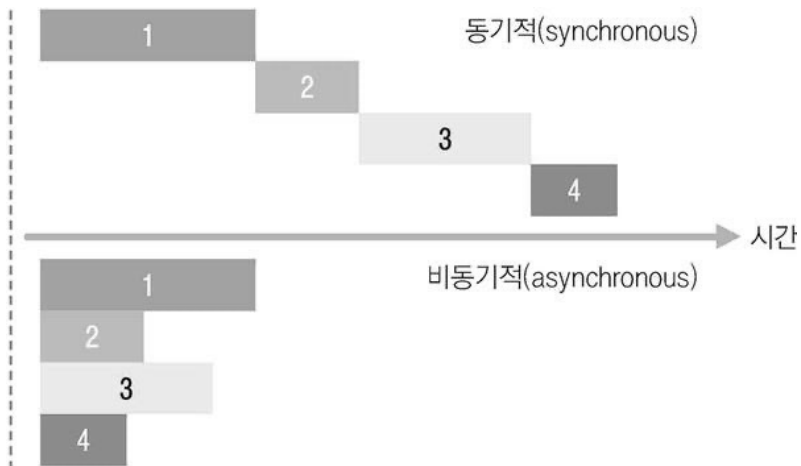
Delete | DELETE

# API 호출

---



## 비동기 작업의 이해



**비동기란?** | 앞선 작업의 마침과 관계없이 다음 동작을 실행할 수 있는 방식

=> 순서대로 x. 동시에 여러가지 요청을 처리할 수도 있고, 기다리는 과정에서 다른 함수도 호출할 수 있다.

갑분 비동기?

서버의 API를 호출함으로써 데이터를 수신할 때,  
**비동기적**으로 처리!

## JS에서 비동기 작업을 할 때 사용하는 방법!

### 콜백 함수

---

1. 다른 함수의 인자로써 이용되는 함수
2. 어떤 이벤트에 의해 호출되어지는 함수

=> 다른 **함수**의 인자로써 넘겨진 후 특정 이벤트에 의해 호출되는 **함수**

```
function callback(print){  
  print();  
}  
  
callback(function(){  
  console.log("Hi~");  
});
```

```
function print(){  
  console.log('Hi~');  
}  
  
setTimeout(print, 3000);
```

**Promise란** | 콜백 지옥 같은 코드가 형성되지 않게 하는 방안으로 도입된 기능

```
function callback(print){  
  print();  
}  
  
callback(function(){  
  console.log("Hi~");  
  callback(function(){  
    console.log("Hi~");  
    callback(function(){  
      console.log("Hi~");  
    })  
  })  
});
```

콜백 지옥

```
function increase(number){
  const promise = new Promise((resolve, reject)=>{
    setTimeout(()=>{
      const result = number + 10;
      if(result>50){
        const e = new Error('NumberTooBig');
        return reject(e);
      }
      resolve(result);
    },1000);
  });
  return promise;
}

increase(0)
  .then(number => {
    console.log(number);
    return increase(number);
  })
  .then(number => {
    console.log(number);
    return increase(number);
  })
  .catch(e => {
    console.log(e);
  })
}
```

promise로 콜백 지옥 해결~

(1) .then : Promise에서의 값 받아옴

(2) `.catch` : 예러가 발생한다면  
알 수 있음

**async/await란** | Promise를 더욱 쉽게 사용할 수 있도록 해주는 문법

```
function increase(number){
  const promise = new Promise((resolve, reject)=>{
    setTimeout(()=>{
      const result = number + 10;
      if(result>50){
        const e = new Error('NumberTooBig');
        return reject(e);
      }
      resolve(result);
    },1000);
  });
  return promise;
}

async function runTasks(){
  try{
    let result = await increase(0);
    console.log(result);
    result = await increase(result);
    console.log(result);
    result = await increase(result);
    console.log(result);
  } catch (e) {
    console.log(e);
  }
}
```

(1) 함수 앞부분에 `async` 키워드 추가  
(2) Promise 앞부분에 `await` 사용  
⇒ Promise가 끝날 때까지 기다리고,  
결과 값을 특정 변수에  
담을 수 있음.

# POSTMAN

---

## POSTMAN이란?

---

Postman은 개발한 API를 테스트하고,  
테스트 결과를 공유하여 API 개발의 생산성을 높여주는 플랫폼

- (1) 목 서버 생성
- (2) GET으로 데이터 send
- (3) RUN

멤버 정보를 담은 목서버  
<https://16b9534b-1b6f-4e0a-bd63-b966d5d571f7.mock.pstmn.io/list>

# 데이터 패칭

---

가보자고~



GET

---

READ

데이터를 읽어들이자!~

## useState란?

```
const [ state, setState ] = useState( );
```



렌더링을  
일으킬 수  
있는 변수



state의 값을  
변경할 때  
사용하는 함수




state의 초기값을  
정할 수 있고,  
return 값으로  
state, setState를  
돌려주는 hook



초기값

## useEffect란?

리액트 컴포넌트가  
렌더링될 때마다  
특정 작업을 수행하도록  
설정할 수 있는 Hook



```
useEffect(() => {  
  console.log("최고");  
}, [ ]);
```

검사하고 싶은 값 삽입  
비워두면 useEffect에서  
설정된 함수가  
처음에만 실행되고 이후 실행x

## useNavigate이란?

```
const navigate = useNavigate();  
const goArticles = () => {  
  navigate('/articles');  
  // articles 페이지로 이동  
}
```



Link 컴포넌트를  
사용하지 않고  
다른 페이지로 이동해야 하는  
상황에 사용하는 Hook

## useLocation이란?

```
const location = useLocation();
```



```
useEffect(() => {  
  console.log(location);  
}, [ location ])
```

현재의 URL을 대표하는  
location 객체를 반환하는 Hook.  
URL이 바뀔 때마다  
새로운 location이 반환되는  
useState같은 Hook

## fetch

---

### fetch란? | JS Web API

HTTP 파이프라인을 구성하는 요청과 응답 등의 요소를  
JS에서 접근하고 조작할 수 있는 인터페이스를 제공

```
fetch('http://example.com/movies.json')  
  .then((response) => response.json())  
  .then((data) => console.log(data));
```



## axios

---

**axios란?** | 현재 가장 많이 사용되고 있는 자바스크립트 HTTP 클라이언트  
HTTP 요청을 promise 기반으로 처리함

**\$ yarn add axios**

```
axios.get("https://16b9534b-1b6f-4e0a-bd63-b966d5d571f7.mock.pstmn.io/list");
```

# 실습 가보자고~

- (1) fetch로 데이터 패칭
- (2) axios로 데이터 패칭
- (3) loading 구현
- (4) 라우팅 적용
- (5) POST 발담그기



```
{members.map((member) => (  
  <Link  
    to={`/${member.memId}`}  
    name={member.name}  
    nickname={member.nickname}  
    description={member.description}  
    role={member.role}  
    <StyledButton  
      onClick={() => handleClick(`/${member.memId}`, member.memId)}  
    >  
      <StyledButton>{member.name}</StyledButton>  
    </Link>  
    {member.name}  
  </StyledButton>  
))}
```

Link로는 props를 보낼 수 없다!

# 백프론트 합체

---

그래서 실전에서는..?

## 백엔드

```
class User(AbstractUser):
    nickname = models.CharField(max_length=20, blank=True, default="")
    grade = models.SmallIntegerField(default=0)
    age = models.SmallIntegerField(default=0)
    profile_photo = models.ImageField(blank=True, null=True, upload_to="accounts_photo/")
    bio = models.CharField(max_length=30, blank=True, default="")
    realname = models.CharField(max_length=15, blank=True, default="")
    major = models.CharField(max_length=15, blank=True, default="")
```

우리는 이런 데이터를 json으로 읽어와서  
사용했지요!

Nickname:	<input type="text" value="노는게젤좋아"/>
Grade:	<input type="text" value="3"/>
Age:	<input type="text" value="0"/>
Profile photo:	Currently: <a href="#">accounts_photo/제목_없음_11.PNG</a> <input type="checkbox"/> Clear Change: <input type="button" value="파일 선택"/> 선택된 파일 없음
Bio:	<input type="text" value="나눔 코딩 감자. 코딩빼고 다 물어봐여"/>
Realname:	<input type="text" value="서지수"/>
Major:	<input type="text" value="경영학부"/>

## 프론트엔드

---

```
useEffect(() => {  
  const fetchData = async () => {  
    setLoading(true);  
    const response = await axios.get(  
      "https://16b9534b-1b6f-4e0a-bd63-b966d5d571f7.mock.pstmn.io/list"  
    );  
    setMembers(response.data);  
    setLoading(false);  
  };  
  fetchData();  
}, []);
```

목 서버를 실제 서버로 변경!



오늘도 과제를 내볼게에

## 필수 과제

---

기한 | ~8/2 (화) 18:00 까지

내용 | 당근마켓의 모든 데이터를 API로 연결하기

\*조건\*

(1) postman의 목서버를 이용해주세요!

<https://www.youtube.com/watch?v=RnpeQd1xJi8>

(2) axios를 이용해주세요!

## 선택(이라 쓰고 필수라 읽는다) 과제

---

내용 | 코드라이언 강의 싹 다 듣기 \*^^\*



집가자