Fetch API

Fetch API는 자바스크립트에서 사용할 수 있는 현대적인 인터페이스로, 비동기적으로 네트워크 통신을 하기 위해 사용됩니다. 이는 XMLHttpRequest (XHR)의 보다 강력하고 유연한 대안으로, 리소스(대개 HTTP API)를 네트워크에서 쉽게 가져올 수 있게 해줍니다.

특징	설명
Promise 기반	Fetch API는 Promise를 반환합니다. 비동기적으로 네트워크 응답을 처리할 수 있으며, 콜백 함수보다 깔끔하고 관리하기 쉬운 코드를 작성할 수 있습니다.
유연한 요청과 응답 처리	Fetch API는 요청 및 응답을 위한 다양한 옵션을 제공합니다. 예를 들어, HTTP 메소드(GET, POST 등), 헤더, 본문 데이터 등을 쉽게 설정할 수 있습니다.
표준화된 접근 방식	Fetch는 모던 웹 개발 환경에서 네트워크 요청을 처리하는 표준 방식으로 자리잡고 있습니다. 대부분의 최신 브라우저에서 지원됩니다.
Response 인터페이스	Fetch API는 응답 데이터를 다루기 위한 Response 인터페이스를 제공합니다. 응답 상태, 헤더 등 다양한 정보에 접근하고, 응답 본문을 다양한 형식(JSON, 텍스트, Blob 등)으로 변환할 수 있습니다.
보안	Fetch API는 같은 출처 정책(Same-Origin Policy)을 따르며, CORS(Cross-Origin Resource Sharing)를 지원하여 보안을 강화하는 데 도움이 됩니다.

- Fetch API

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

---

fetch의 기본 구조

`npm install node-fetch` 설치
`"type": "module"` 추가 (package.json)
`node app.js` 실행

```
import fetch from 'node-fetch';

// 테스트 데이터 주소 : https://api.thedogapi.com/v1/breeds
// 테스트 데이터 이미지 주소 : https://api.thedogapi.com/v1/images/BJa4kxc4X

// Fetch 함수를 호출하여 웹에서 데이터를 불러옵니다.
fetch('https://api.thedogapi.com/v1/breeds')
  .then(response => {
```

```
    // 응답이 성공적인지 확인합니다.
    if (!response.ok) {
      // 성공적이지 않으면, 오류를 던집니다.
      throw new Error('네트워크 응답이 올바르지 않습니다.');
    }
    // 응답을 JSON 형태로 파싱합니다.
    return response.json();
  })
  .then(async data => {
    // [1] 데이터 정상 확인 (파싱된 데이터를 출력)
    console.log(data);

    // [2] 각 요소에 해당하는 모든 키의 값 출력 (각 요소의 name 만 출력)
    const names = data.map(item => item.name);
    console.log(names);

    // [3] 각 요소에서 id와 name을 추출하여 새로운 객체 반환
    const idNamePairs = data.map(item => ({ id: item.id, name: item.name
}));
    console.log(idNamePairs);

    // [4-1] 각 요소에 해당하는 이미지 주소 출력 (async await)
    let images = []; // 이미지 URL을 저장할 배열을 초기화하고, 각 개 종의 이미지를 가져와
서 배열에 추가합니다.
    for (const item of data) { // 각 강아지 종의 정보와 이미지 URL을 가져오기 위해 반
복문을 사용합니다.
      const result = {
        id: item.id,     // 현재 강아지 종의 ID를 결과 객체에 추가합니다.
        name: item.name, // 현재 강아지 종의 이름을 결과 객체에 추가합니다.
        // 강아지 종의 이미지 URL을 가져오는 getImageFromImageId 함수를 호출하고, 해당
URL을 결과 객체에 추가합니다.
        // getImageFromImageId 함수는 강아지 종의 reference_image_id를 사용하여 이
미지 정보를 가져옵니다.
        image_url: await getImageFromImageId(item.reference_image_id) //
await 할 때 한번 멈추기 때문에 순차적으로 실행됨 (동기 실행)
      }
      console.log(result);
      images.push(result); // images 배열에 강아지 종의 이미지 정보를 추가합니다.
    }
    console.log(images); // images 배열을 출력하여 이미지 정보를 확인합니다.

    // [4-2] 각 요소에 해당하는 이미지 주소 출력 (Promise)
    const promises = data.map(item => new Promise((resolve, reject) => {
      getImageFromImageId(item.reference_image_id)
        .then((image_url) => {
          resolve({
            id: item.id,     // 현재 강아지 종의 ID를 결과 객체에 추가합니다.
            name: item.name, // 현재 강아지 종의 이름을 결과 객체에 추가합니다.
            // 강아지 종의 이미지 URL을 가져오는 getImageFromImageId 함수를 호출하고,
해당 URL을 결과 객체에 추가합니다.
            // getImageFromImageId 함수는 강아지 종의 reference_image_id를 사용하
여 이미지 정보를 가져옵니다.
            image_url: image_url
          })
```

```
        })
    }))
    return Promise.all(promises); // 비동기로 여러개가 한번에 실행되서 promise가 반환
됨
    // Promise.all
  })
  // Promise.all 에서 반환된 값을 array로 반환
  .then((dogs) => {
    console.log(dogs); // images 배열을 출력하여 이미지 정보를 확인합니다.
  })
  .catch(error => {
    // 요청 중에 오류가 발생하면 여기에서 처리합니다.
    console.error('Fetch 요청 중 오류 발생:', error);
  });

// 각 강아지 종의 이미지 주소를 가져오는 함수
async function getImageFromImageId(referenceImageId) {
  try {
    // 강아지 종의 이미지 정보를 가져옵니다.
    const breedImageResponse = await
fetch(`https://api.thedogapi.com/v1/images/${referenceImageId}`);
    const breedImageData = await breedImageResponse.json();

    // 강아지 종의 정보와 이미지 URL을 반환합니다.
    return breedImageData.url;
  } catch (error) {
    // 오류가 발생하면 콘솔에 오류 메시지를 출력합니다.
    console.error('오류 발생:', error);
    return null;
  }
}
```