2. HTTP요청 _GET

> - GET Request
> - 비동기(콜백, 프로미스, async/await)

1. GET 요청을 직접 보내보기

- 필요한것 : URL, (브라우저에서 제공하는) fetch API, (선택) header 등 다른 옵션들
- `practice 4_15`

```
import React, { useEffect, useState } from 'react';
import NewsItem from '../NewsItem';

const NewsList = () => {
    const [news, setNews] = useState([]);

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

    const getNewsList = () => {
        fetch('https://newsapi.org/v2/everything?q=tesla&from=2023-10-
17&sortBy=publishedAt&apiKey=f9b8fbf5ae5348fa8d65005595d297d7e')
            .then((response)  => response.json())
            .then((data) => {
            console.log(data);
            setNews(data.articles);
    )};

    getNewsList();

    const dummy =[...]
    return(
                <div>{dummy.map(news => (
                        <NewsItem {...news} />
                    ))}</div>
            );
};

export default NewsList;
```

```
//NewsItem
import React from 'react';
import {NewsItemWrapper} from './style.js';

const NewsItem = (props) => {
    return(
        <NewsItemWrapper>
            <h1>{props.title}</h1>
            <div className="content">
```

```
                <h3>{props.description}</h3>
                <img className="image" src={props.urlToImage} alt=""/>
            </div>
        </NewsItemWrapper>
    );
};

export default NewsItem;
```

- **Fetch API**

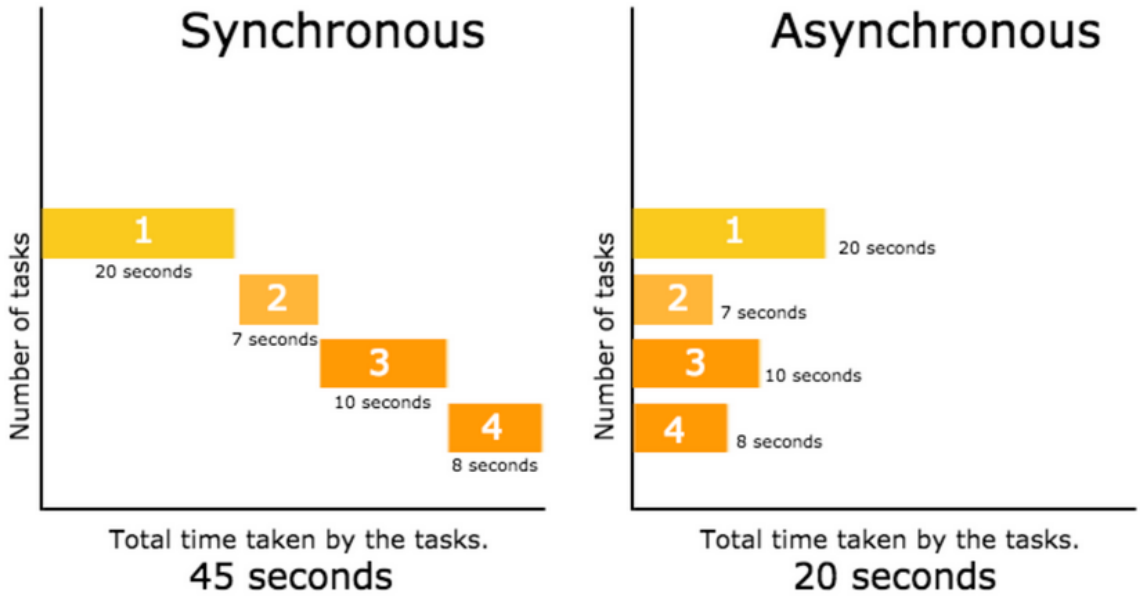
Fetch API 는 네트워크 통신을 포함한 리소스 취득을 위한 인터페이스를 제공하며 [XMLHttpRequest] 보다 강력하고 유연한 대체제 이다. (인터페이스는 추상화된 도구라고 생각하면 된다.)

```
async function logJSONData() {
  const response = await fetch("http://example.com/movies.json");
  const jsonData = await response.json();
  console.log(jsonData);
}
```

`fetch()` 메서드는 하나의 필수 매개변수로 가져오려는 리소스 경로를 받습니다. 반환 값은 해당 요청에 대한 Response로 이행하는 Promise인데, 서버가 헤더를 포함한 응답을 하는 순간 이행합니다. 이는 서버가 HTTP 오류 응답 코드로 응답해도 이행한다는 뜻이다. 선택사항으로 두번째 매개변수에 `init` 옵션 객체를 제공 할 수 있습니다.

2. 비동기

하나의 프로세스가 완료되기 전에 다음 프로세스를 시작하는 방식




```
function func1() {
    console.log('func1');

    func2()
}
```

```
function func2() {
    setTimeout(() => {
        console.log('func2');
    }, 0);

    func3();
}

function func3() {
    console.log('func3');
}

func1();
```

🤔 콘솔은 어떻게 찍힐까?

```
func1
func3
func2
```

브라우저 런타임, 브라우저 런타임에 여러가지 요소들이 있는데, 대표적으로 자바스크립트 엔진이 있고 실행컨텍스트도 있고 힙도 있고 바깥쪽으로 web API도 있고 콜백 큐, 이벤트 루프도 있다.

콜백함수와 같이 어떤 비동기 통신이 일어날때는 그 함수를 콜백 큐로 보낸 다음에 실행 컨텍스트가 비어있으면, 콜백 큐에서 하나씩 빼서 실행컨텍스트에 넣어서 실행을 한다. 그리고 그 과정에서 하나씩 빼내는 역할이 이벤트 루프이다.

콜백 큐에 대해 리마인드 하면서 학습 이어나가기.

---

(1) `func1();` 이 실행이 된다.

(2) `func2()` 는 `setTimeout` 이 실행이 된다. `setTimeout` 은 기본적으로 콜백함수이다. 따라서 다음 코드를 콜백 큐로 넣어준다. 넣어준 다음에는 아래쪽 코드가 실행이 된다.

```
//콜백함수
() => {
        console.log('func2');
    }
```

=> `console.log('func2');` 를 콜백큐로 넣고 `func3();` 를 실행컨텍스트에 넣는다.

(3) 실행컨텍스트에서 `func3();` 가 찍힌다 `func3();` 가 빠져 나가면 실행 컨텍스트는 비어지게 된다.

(4) 이때 콜백큐에 있었던 콜백함수가 뽑아져 나와서 실행컨텍스트에 들어가고 그 다음에 `console.log('func2')` 가 찍히게 되는 것이다.

---

(1) 콜백함수

`setTimeout()` 같이 특정 함수 안에 매개변수 형태로 전달 받은 함수를 의미

```
function Callback(cb) {
    console.log('콜백 함수 안');
    cb();
}
Callback(() => console.log('이 부분이 콜백'))
```

(2) Promise

ES6에서 등장한 자바스크립트에서 비동기를 처리하기 위해 사용하는 객체

: 나는 너에게 데이터를 줄 약속을 할게

`Promise` 가 생성된 시점에는 알려지지 않았을 수도 있는 값을 위한 대리자로, 비동기 연산이 종료된 이후에 결과 값과 실패 사유를 처리하기 위한 처리기를 연결할 수 있다. 프로미스를 사용하면 ⭐**비동기 메서드에서 마치 동기 메서드처럼 값을 반환할 수 있다.** 다만 최종 결과를 반환하는 것이 아니고, 미래의 어떤 시점에 결과를 결과를 제공하겠다는 약속을 반환한다.

- 대기(pending) : 이행하기도, 거부하지도 않은 초기 상태
- 이행(fulfilled) : 연산이 성공적으로 완료됨
- 거부(rejected) : 연산이 실패함

```
const promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        reject(new Error());
    }, 1000);
});

myPromise
    .then(n => {
        console.log(n);
    })
    .catch(error => {
        console.log(error);
    });
```

(3) ⭐ Async/await

자바스크립트에서 promise를 가지고 비동기를 동기처럼 사용하는 문법

```
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}

async function process(){
    console.log('안녕하세요!');
    await sleep(1000); //1초 쉬고
    console.log('반갑습니다!');
}

process();
```

`sleep` 프로미스를 반환하는 함수

⭐ `async` 키워드로 함수를 감쌌을때만 그 안에 `await`를 사용할 수 있다.

- fetch 코드를 async/await 함수로 바꿔주기

```
    const getNewsList = async () => {
        const response = await fetch('https://newsapi.org/v2/everything?
q=tesla&from=2023-10-
17&sortBy=publishedAt&apiKey=f9b8fbf5ae5348fa8d6500595d297d7e');
        const data = await response.json();
            setNews(data.articles);
    )}
```

- ⭐HTTP GET 요청을 통해 서버에서 데이터를 불러올 수 있고, 이 방식은 비동기적으로 일어난다.
- ⭐자바스크립트에서 비동기를 처리하는 방식은 3가지 이다. (콜백함수, promse, asyc/awaits)