3. HTTP요청_로딩 및 에러 처리

1. Loading 처리

클라이언트에서 서버에 요청을 보내고, 그 요청을 응답받기까지는 시간이 걸린다 (일반적으로 수십 ms, 하지만 데이터 크기 및 네트워크 환경에 따라 오래 걸리기도 한다.)

```
const [isLoading, setIsLoading] = useState(false);
```

> 현재는 로딩을 하고 있지 않기 때문에 초기값을 `false` 로 설정해준다.

```
return(
    <div>
        {!isLoading && news.map((news) => <NewsItem {...news} />)}
        {isLoading && <p>로딩중입니다.</p>}
    </div>
)
```

```
const getNewsList = () => {
        setIsLoading(true);
        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);
        setIsLoading(false);
)};
```

2. Error 처리

서버로부터 응답이 항상 성공적으로 오는 것은 아니다. 응답에 상태 코드(state code)가 있는데, 이에 따라 응답이 성공인지 실패인지 알 수 있다.

200	401	403	404	500
success	unauthorized	forbidden	not found	internal server error

서버와 요청, 응답을 주고 받는 코드를 `try { } ... catch { }` 로 감싸준다.

```
try {
    setIsLoading(true);
        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);
        setIsLoading(false);
} catch(error) {
    console.error(error.message)
}
```

- 요청을 보내고 응답을 받는 사이 비동기 로직은 시간이 소요되므로 적절한 로딩 처리가 필요하다.
- 서버로부터 응답이 항상 성공적으로 오지 않으므로 요청을 보내는 부분은 `try..catch`로 감싸줘야 한다.