## 2. 커스텀 훅 (2)

- HTTP 커스텀 훅

: 앞으로 백엔드와 협업해서 프로젝트를 하게 될 경우 HTTP 요청을 보내게 될 일이 많다. 그때마다 매번 fetch로 하나씩 요청을 보낼 것인가? Loading 처리, error 처리 등 매번 해부는 건 상당히 비효율적이다.

- `practice-4_19`

```
// hooks 폴더 생성
// useHttp.js 파일 생성

import React from 'react';

const useHttp = (requestConfig, applyData) => {
    const [isLoading, setIsLoading] = useState(false);
    const [error,setError] = useState(null);

    const sendRequest = async () => {
        setIsLoading(true);
        setError(null);
        try{
            const response = await fetch(
                requestConfig.url, {
                    //필요한 경우에만 작성하기
                    //삼항연산자
                    method: requestConfig?.method ?  requestConfig?.method:
'GET',
                    headers: requestConfig?.headers ? requestConfig?.headers :
{},
                    body: requestConfig?.body ? requestConfig?.body : null
                })
            );

            if(!response.ok) {
                throw new Error("Request failed!");
            }
            const data = await response.json();

            applyData(data);
        }catch(err) {
            setError(err.message || "Somthing went wrong")
        }
        setIsLoading(false);
    };
    return {
        isLoading,
        error,
        sendLoading
    };
};

export default useHttp;
```

```
import useHttp from './hooks/useHttp';

const { isLoading, error, sendRequest: fetchTasks } = useHttp({
    url: '_____________',
}, transformTasks );
```

---

- HTTP요청과 같이 프로젝트 전체적으로 자주 쓰이는 로직들은 아웃소싱 해서 커스텀 훅으로 사용하면 생산성이 올라갈 수 있다.