useEffect
&
JSON

Fetch ~ then ~ catch ~ finally

➢ Fetch(url)~ .then() ~.catch()~ .finally() 구조

① fetch(url) 은 네트워크 요청을 보내고 나중에(비동기) 응답을 받을 때까지 기다리는 Promise를 반환

② 그 Promise가 "이행(fulfilled)"되면 .then() 안의 콜백이 실행

③ 실패하면 .catch()가 실행

④ .finally()는 성공/실패에 관계없이 항상 실행

```
fetch('https://.../users')
```

- fetch는 브라우저 내장 함수로 네트워크 요청(HTTP GET 등)을 보낸다.
- 즉시 Promise를 반환: Promise<Response> 타입 (응답 객체를 나중에 준다).
- 네트워크가 느리면 브라우저는 요청을 백그라운드로 보내고 다음 코드를 바로 실행한다(비동기).

Fetch ~ then ~ catch ~ finally

then((res) => res.json())

- 첫 번째 .then의 파라미터 res 는 Response 객체이다.
  Response 객체에는:
  - res.status (HTTP 상태코드, 예: 200)
  - res.ok (응답이 200~299이면 true)
  - res.headers, res.text(), res.json() 등 메서드가 있음
  - 중요: res.json() 은 동기 함수가 아님 — 이것도 Promise를 반환 → JSON 텍스트를 파싱해서 자바스크립트 객체(또는 배열)로 변환하는 작업이 비동기이기 때문.
- 그래서 .then(res => res.json()) 는 다음 .then에 실제 데이터(파싱된 JS 객체)를 넘겨줌

..then((data) => { ... })

- 이 data는 res.json()이 파싱해서 반환한 자바스크립트 값(여기선 사용자 배열)
- 이 블록 안에서 실제 데이터를 다루면 됨
- console.log('가져온 데이터:', data) — 개발용 로그
- data.filter(...) — 배열 필터링
- setResults(filtered) — React state 업데이트

Fetch ~ then ~ catch ~ finally

.catch((err) => console.error('에러 발생:', err))

- fetch 자체가 네트워크 실패(인터넷 끊김 등)로 거부되면 이 블록이 호출됨
- 또한 .then() 내부에서 throw를 하거나, res.json()이 파싱 실패하면 이 .catch로 들어온다.
- 주의: HTTP 404/500 같은 응답은 fetch가 자동으로 에러로 처리하지 않는다
- (응답은 성공적으로 도착했기 때문).
- 그러므로 HTTP 에러는 직접 검사해서 throw 해야 한다.

.finally(() => { setLoading(false); })

- 성공/실패에 관계없이 항상 실행됨
- UI에서 "로딩 상태"를 해제할 때 매우 유용하다.
- 예: 네트워크 성공 시에도, 실패 시에도 loading을 false로 바꿔서 스피너를 숨기는 용도.

➢ Error 코드를 작성하는 이유

- 네트워크 통신은 예측할 수 없는 상황(에러)이 자주 발생한다.

  예를 들어,

  - 인터넷이 끊겼거나,
  - 서버가 응답하지 않거나,
  - 응답은 왔지만 형식이 잘못되었을 수도 있다.

- 그래서 개발자는 이런 오류가 발생하더라도 프로그램이 멈추지 않도록 에러를 미리 감지하고 안전하게 처리하는 코드(fetch의 .catch() 등)를 작성해야 한다.

- 이것이 네트워크 코드를 작성할 때의 기본 원칙이다.

Fatch ~ then 에러 없이 정상 작동

```jsx
export default function ExFatch() {
  // 1. 상태 변수들
  const [loading, setLoading] = useState(true); // 로딩중 표시
  const [data, setData] = useState([]); // 가져온 데이터 저장
  const [error, setError] = useState(null); // 에러 메시지 저장

  useEffect(() => {
    console.log('데이터 요청 시작');
    fetch('https://jsonplaceholder.typicode.com/users')
      .then((res) => {
        if (!res.ok) {
          // HTTP 상태 코드 확인
          throw new Error(`HTTP error! status: ${res.status}`);
        }
        return res.json(); // JSON 파싱 (Promise)
      })
      .then((result) => {
        console.log('가져온 데이터:', result);
        setData(result); // 상태 저장
      })
      .catch((err) => {
        console.error('X 에러 발생:', err);
        setError(err.message); // 에러 상태 저장
      })
      .finally(() => {
        console.log('요청 완료');
        setLoading(false); // 로딩 종료
      });
  }, []);
}
```

```
데이터 요청 시작
가져온 데이터: (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…
  ▸ 0: {id: 1, name: 'Leanne Graham', username: 'Bret
  ▸ 1: {id: 2, name: 'Ervin Howell', username: 'Anton
  ▸ 2: {id: 3, name: 'Clementine Bauch', username: 'S
  ▸ 3: {id: 4, name: 'Patricia Lebsack', username: 'K
  ▸ 4: {id: 5, name: 'Chelsey Dietrich', username: 'K
  ▸ 5: {id: 6, name: 'Mrs. Dennis Schulist', username
  ▸ 6: {id: 7, name: 'Kurtis Weissnat', username: 'El
  ▸ 7: {id: 8, name: 'Nicholas Runolfsdottir V', user
  ▸ 8: {id: 9, name: 'Glenna Reichert', username: 'De
  ▸ 9: {id: 10, name: 'Clementina DuBuque', username:
    length: 10
  ▸ [[Prototype]]: Array(0)
요청 완료
```

기본으로 존재하는 표준 내장 속성이다.

Fatch ~ then 에러

```jsx
export default function ExFatch() {
  // 1. 상태 변수들
  const [loading, setLoading] = useState(true); // 로딩중 표시
  const [data, setData] = useState([]); // 가져온 데이터 저장
  const [error, setError] = useState(null); // 에러 메시지 저장

  useEffect(() => {
    console.log('데이터 요청 시작');
    fetch('https://jsonplaceholder.typicode.com/userss')
      .then((res) => {
        if (!res.ok) {
          // HTTP 상태 코드 확인
          throw new Error(`HTTP error! status: ${res.status}`);
        }
        return res.json(); // JSON 파싱 (Promise)
      })
      .then((result) => {
        console.log('가져온 데이터:', result);
        setData(result); // 상태 저장
      })
      .catch((err) => {
        console.error('X 에러 발생:', err);
        setError(err.message); // 에러 상태 저장
      })
      .finally(() => {
        console.log('요청 완료');
        setLoading(false); // 로딩 종료
      });
  }, []);
}
```

<p> 태그에 출력

X 에러: HTTP error! status: 404

react-dom_client.js

Download the React DevTools for a bette
experience: https://react.dev/link/reac

데이터 요청 시작

▶ GET
https://jsonplaceholder.typicode.com/userss

▶ X 에러 발생: Error: HTTP error! status: 4
at ExFatch.jsx:16:17

요청 완료

Async ~ await 에러 처리 방식

```
useEffect(() => {
    async function fetchData() {
      try {
        setLoading(true); // 로딩 시작
        const res = await fetch('https://jsonplaceholder.typicode.com/users');
        if (!res.ok) throw new Error(`HTTP error! status: ${res.status}`);
        const data = await res.json(); // JSON 파싱
        console.log('가져온 데이터:', data);
      } catch (err) {
        console.error('에러 발생:', err);
      } finally {
        setLoading(false); // 로딩 종료
      }
    }
    fetchData(); // 비동기 함수 호출
  }, []);
```

예제1] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://jsonplaceholder.typicode.com/users 이용해 JSON 데이터를 가저오세요

② 특정 사용자 정보만 보도록 작성하시오.

③ useEffect의 의존성 배열 사용해 작성 하세요

④ userId 상태값을 추가하고, userId가 변경될 때마다 해당 사용자의 정보(/users/${userId})만 가져와 화면에 출력되도록 작성하시오.

⑤ 함수 이름은 ExJ01.jsx로 작성하시오.

⑥ src 폴더 안에 JSON 폴더 -> ExJ01.jsx 파일을 작성하시오.

⑦ src/App.jsx에서 ExJ01.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

<다음 사용자 보기 클릭 한 후>

예제2] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://jsonplaceholder.typicode.com/posts 이용해 JSON 데이터를 가저오세요

② 게시글 데이터 불러오기를 작성한다.

③ /posts에서 게시글 제목 5개만 가져와 로 화면에 출력한다.

④ 제목 클릭 시 alert로 "내용(content)" 출력되도록 작성하시오.

⑤ 함수 이름은 ExJ02.jsx로 작성하시오.

⑥ src 폴더 안에 JSON 폴더 -> ExJ02.jsx 파일을 작성하시오.

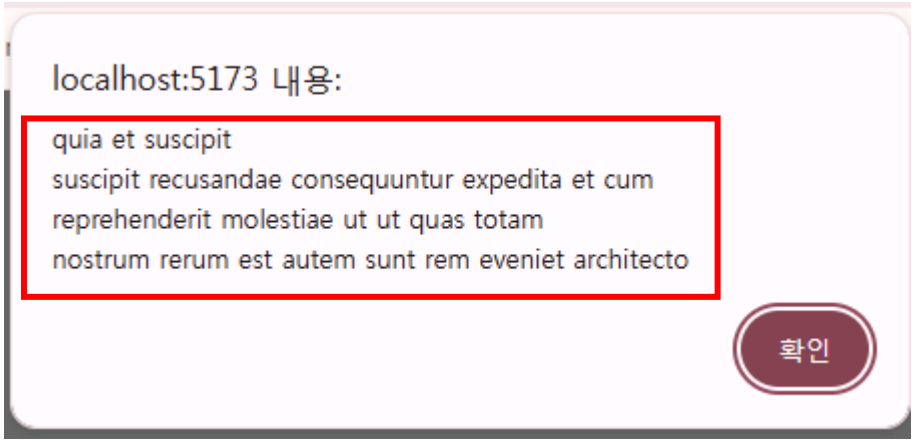
⑦ src/App.jsx에서 ExJ02.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

게시글 5개

- sunt aut facere repellat provident occaecati excepturi optio reprehenderit
- qui est esse
- ea molestias quasi exercitationem repellat qui ipsa sit aut
- eum et est occaecati
- nesciunt quas odio

<각 게시글 클릭 후 내용 alert 으로 띄우기>



예제3] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://jsonplaceholder.typicode.com/posts 이용해 JSON 데이터를 가저오세요

② fetch 실패 시 에러 메시지 출력되도록 작성하시오.

③ 잘못된 주소(예: /postss)로 요청한다.

④ Try~catch 또는 .catch()를 이용해 오류를 출력한다.

⑤ 에러 발생 시 "데이터 요청 실패!" 를 화면에 출력한다.

⑥ 함수 이름은 ExJ03.jsx로 작성하시오.

⑦ src 폴더 안에 JSON 폴더 -> ExJ03.jsx 파일을 작성하시오.

⑧ src/App.jsx에서 ExJ03.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>


데이터 요청 실패!
react-dom_client.js?v
Download the React DevTools for a better devel
experience: https://react.dev/link/react-devtc
GET https://jsonplaceholder.typicode.com/postss
404 (Not Found)

예제4] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://jsonplaceholder.typicode.com/users 이용해 JSON 데이터를 가저오세요

② JSON 데이터와 컴포넌트 분리하여 작성한다.

③ Props로 데이터 전달하여 작성한다.

④ UserList 컴포넌트를 만들어 users 데이터를 props로 전달한다.

⑤ UserList에서 map으로 렌더링하여 작성한다.

⑥ 함수 이름은 ExJ04.jsx로 작성하시오.

⑦ src 폴더 안에 JSON 폴더 -> ExJ04.jsx 파일을 작성하시오.

⑧ src/App.jsx에서 ExJ04.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

컴포넌트 분리 연습

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

예제5] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

① https://jsonplaceholder.typicode.com/users 이용해 JSON 데이터를 가저오세요

② API 호출 시 로딩 스피너 구현하도록 작성하시오.

③ 로딩 중에는 "불러오는 중..." 표시되도록 작성하시오.

④ 3초 후가 지나면 목록의 이름이 화면에 의 로 출력되도록 작성하시오.

⑤ 함수 이름은 ExJ05.jsx로 작성하시오.

⑥ src 폴더 안에 JSON 폴더 -> ExJ05.jsx 파일을 작성하시오.

⑦ src/App.jsx에서 ExJ05.jsx을 import하여 실행 결과를 확인하시오.

<출력 결과>

불러오는 중...

<3초 후 출력 결과>

- Leanne Graham
- Ervin Howell
- Clementine Bauch
- Patricia Lebsack
- Chelsey Dietrich
- Mrs. Dennis Schulist
- Kurtis Weissnat
- Nicholas Runolfsdottir V
- Glenna Reichert
- Clementina DuBuque

예제6] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

https://picsum.photos/

① https://jsonplaceholder.typicode.com/photos? 이용해 JSON 데이터를 가저오세요

② ProductApp.jsx, ProductItem.jsx, ProductList.jsx, ProductApp.css 로 컴포넌트를 분리하여 작성하시오. (단, css 파일도 별도로 작성하여 스타일을 지정한다.)

③ 위에 제시된 JSON URL에서 id, title, img 를 가지고와 화면에 출력하시오.
(단, 위 JSON URL은 가격이 존재하지 않아 가격은 임의대로 만들어 사용 할 것)

④ JSON에서 가져오는 데이터의 개수는 20개로 정하여 작성한다.

⑤ 그 이외의 나머지는 각자 개인이 알아서 작성한다.

⑥ src 폴더 안에 JSON 폴더 -> ExJ06 폴더 -> ProductApp.jsx, ProductItem.jsx, ProductList.jsx, ProductApp.css 파일을 작성하시오.

⑦ src/App.jsx에서 ProductApp.jsx 을 import하여 실행 결과를 확인하시오.

<출력 결과>

상품 목록

accusamus beata

11,200원

officia porro i

13,600원

accusamus ea al

17,200원

reprehenderit e

12,400원

culpa odio esse

14,800원

officia delectu

18,400원