프론트엔드에서의
에러핸들링

이은규

소프트웨어의 오류

컴파일 에러

```
// 생략된 세미콜론으로 인한 컴파일 에러
const greeting = "Hello, World"
console.log(greeting)

[1, 2, 3].forEach(num => console.log(num))
```

'SyntaxError'

```
const greeting = "Hello, World"[1, 2, 3].forEach(num => console.log(num
```

코드가 컴파일 될 때 컴파일러가 해석하지 못해서 발생

런타임 에러

```
let numbers = [1, 2, 3];

// 런타임 에러: 배열 인덱스 범위 초과
console.log(numbers[5]);
```

프로그램이 동작할 때 발견할 수 있는 에러

JavaScript & TypeScript

- JavaScript 언어는 Dynamic Typed Language이기 때문에 **프로그램이 동작할 때 실시간으로 Type이 결정**
- 다른 정적언어라면 코드를 작성하며 잡을 수 있는 에러도 최악의 경우 사용자가 어플리케이션을 이용할 때 에러가 발생

JavaScript(Dynamically Typed Language) : 모든 에러가 컴파일 단계가 아닌 런타임 환경에서 발생

TypeScript(Statically Typed Language): 컴파일 환경에서 타입에 관련된 에러를 잡을 수 있음.

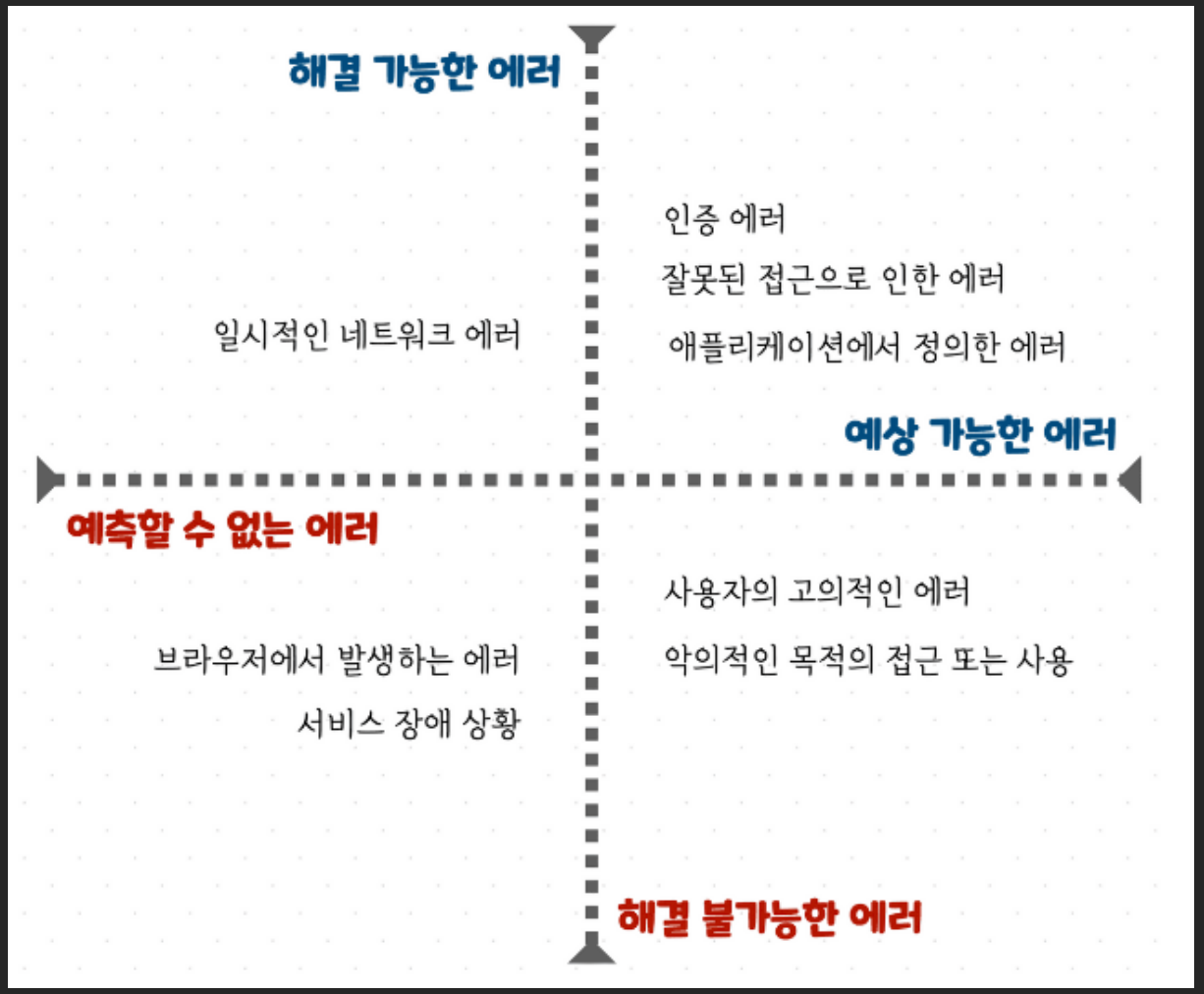
JavaScript 에서의 에러 핸들링

- JavaScript에서는 예외가 발생하면 에러 객체를 내보내고 처리를 하지 않았다면 프로그램이 종료됨.
- 예외로 인해서 발생한 에러 객체를 핸들링하는 것을 **에러 핸들링** 이라고 함.

에러 핸들링이 필요한 이유

- 에러에 대해 사용자에게 인지시키고 다른 행동을 하도록 유도하여 서비스에 대한 부정적인 경험을 막을 수 있음
- 정확한 에러 핸들링을 하지 않는다면 단순히 정보를 요청하는 것이 아니라 데이터베이스에 정보를 넣는 등의 작업일 경우 서비스의 트랜잭션에 영향을 미쳐 서비스 장애를 일으킬 수 있는 상황까지 이어질 수 있음.

에러의 분류


해결 가능한 에러
인증 에러
잘못된 접근으로 인한 에러
애플리케이션에서 정의한 에러
일시적인 네트워크 에러
예상 가능한 에러
예측할 수 없는 에러
사용자의 고의적인 에러
악의적인 목적의 접근 또는 사용
브라우저에서 발생하는 에러
서비스 장애 상황
해결 불가능한 에러

예측이 가능한 에러

- 인증 에러
- 없는 페이지를 접근했을 때의 에러
- 어플리케이션에서 정의한대로 API 응답의 상태 코드로 예측할 수 있는 에러
- 악의적인 목적으로 접근했을 때 이를 보완하는 코드가 프로그램에 내제되어 있지 않은 경우

예측이 불가능한 에러

- 서비스 장애
- 너무 많은 인원이 몰려 일시적인 네트워크가 불안정한 상황에서 발생하는 에러
- 500대 에러

Try – Catch - Finally

```
const crewNickname = "티케타카";

openStudyLog();
try {
  writeStudyLog(crewNickname);
} catch (error) {
  console.error("등록된 크루가 아닙니다.");
} finally {
  closeStudyLog();
}
```

Try

에러가 발생할 수도 있는 로직 작성
에러를 직접 throw 할 수 있음

Catch

에러 잡기

Finally

로직의 성공 & 실패 여부와 상관없이 무조건 실행되는 부분

무조건 Try – Catch는 지양

1. Catch에 넘겨지는 에러 객체의 타입을 보장할 수 없다.

- Catch에 넘어가는 에러가 분기처리를 하지 않으면 무엇인지 명확히 알 수 없다.

2. 오류를 해결하는 것이 아닌 숨기는 것.

- 모든 예외 사항에 대해서 Catch를 했을 경우 사용자에게는 에러가 보이지 않겠지만 이는 오류를 해결하는 것이 아닌 숨기는 것에 불과하다.

만약 에러를 분리하기 위해서 Catch 안에서 Rethrow를 하거나 중첩 Try-Catch를 하게 되면 나중에 어플리케이션이 꺼졌을 때도 코드의 가독성은 훨씬 떨어지게 된다.

에러 핸들링 예시

1. 비동기 API 통신에서의 에러 핸들링
2. 잘못된 페이지나 없는 페이지를 접근
3. 예상할 수 없는 에러 - 모니터링

출처 :

[10분 테코톡] 🌼 티케의 프론트엔드에서의 에러 핸들링
(https://www.youtube.com/watch?v=FXtooPhupr4&list=WL&index=37)

참조 :

클라이언트의 사용자 중심예외 처리, 코어자바스크립트 'try-catch'와 에러 핸들링
https://jbee.io/react/error-declarative-handling-2/
에러처리를 어떻게 하면 좋을까
https://rinae.dev/posts/hot-tohandle-errors-1/
비동기 프로그래밍과 실전 에러 핸들링
https://www.youtube.com/watch?v=o9JnT4sneAQ
프론트엔드 웹서비스에서 우아하게 비동기 처리하기
https://www.youtube.com/watch?v=FvRtoViujGg