Ch47 에러 처리

try ... catch ... finally 문

에러 처리를 구현하는 방법은 크게 두 가지가 있다.

1. `querySelector`나 `Array#find` 메서드처럼 예외적인 상황이 발생하면 반환하는 값을 if 문이나 단축 평가 또는 옵셔널 체이닝 연산자를 통해 확인해서 처리
2. 에러 처리 코드를 미리 등록해 두고 에러가 발생하면 에러 처리 코드로 점프하도록 함

try ... catch ... finally 문은 두 번째 방법이며 finally문은 생략할 수 있지만 catch문은 생략할 경우 try문의 의미가 없어지므로 생략하지 않는다.

```
try {
    // 실행할 코드(에러가 발생할 가능성이 있는 코드)
} catch (err) {
    // try 코드 블록에서 에러가 발생하면 이 코드 블록의 코드가 실행된다.
    // err에는 try 코드 블록에서 발생한 Error 객체가 전달된다.
} finally {
    // 에러 발생과 상관없이 반드시 한 번 실행된다.
}
```

Error 객체

생성자 함수	인스턴스
Error	일반적 에러 객체
SyntaxError	자바스크립트 문법에 맞지 않는 문을 해석할 때 발생하는 에러 객체
ReferenceError	참조할 수 없는 식별자를 참조했을 때 발생하는 에러 객체
TypeError	피연산자 또는 인수의 데이터 타입이 유효하지 않을 때 발생하는 에러 객체
RangeError	숫자값의 허용 범위를 벗어났을 때 발생하는 에러 객체
URIError	encodeURI 또는 decodeURI 함수에 부적절한 인수를 전달했을 때 발생하는 에러 객체
EvalError	eval 함수에서 발생하는 에러 객체

throw 문

- Error 생성자 함수로 여러 객체를 생성한다고 에러가 발생하는 것은 아니다.
- 에러를 발생시키려면 try 코드 블록에서 throw 문으로 에러 객체를 던져야한다.
- throw 문의 표현식은 일반적으로 에러 객체를 지정한다.
- 에러를 던지면 catch 문의 에러 변수가 생성되고 던져진 에러 객체가 할당된 뒤 catch 코드 블록이 실행되기 시작한다.

```
try {
    // 에러 객체를 던지면 catch 코드 블록이 실행되기 시작한다.
    throw new Error('something wrong');
} catch (error) {
    console.log(error);
}
```

에러의 전파

에러는 콜 스택의 아래 방향(호출자 방향)으로 전파된다.

```
const foo = () => {
    throw Error('foo에서 발생한 에러');
};

const bar = () => {
    foo();
};

const baz = () => {
    bar();
};

try {
    baz();
} catch (err) {
    console.error(err);
}
```

foo 실행 컨텍스트(에러 발생) → bar 실행 컨텍스트 → baz 실행 컨텍스트 → 전역 실행 컨텍스트 (에러 전파)

✔️ 비동기 함수인 `setTimeout`이나 `프로미스 후속 처리` 메서드의 콜백 함수는 호출자가 없다는 것에 주의!

(에러를 전파할 호출자가 존재하지 않음)