CHAPTER 08

예외 처리

자바스크립트 프로그래밍 입문 (2판)

Contents

학습목표

- 예외가 무엇인지 이해합니다.
- 예외를 처리하는 기본 방법과 고급 방법을 익힙니다.
- 예외 객체를 활용하는 방법을 익힙니다.
- 예외를 강제로 발생시키는 방법을 익힙니다.

내용

- 예외와 기본 예외 처리
- 고급 예외 처리
- 예외 객체
- 예외 강제 발생

1. 예외와 기본 예외 처리

- 예외 : 실행에 문제가 발생하면 자동 중단됨. 이렇게 발생한 오류
- 예외 처리 : 오류에 대처할 수 있게 하는 것

```
error.error.error()
```

실행 결과

```
C:\example\error.js:1
error.error.error();
^

ReferenceError: error is not defined
    at Object.<anonymous> (C:\example\error.js:1:1)
[90m    at Module._compile (internal/modules/cjs/loader.js:1085:14)[39m
[90m    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)[39m
[90m    at Module.load (internal/modules/cjs/loader.js:950:32)[39m
[90m    at Function.Module._load (internal/modules/cjs/loader.js:790:14)[39m
[90m    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:
76:12)[39m
[90m    at internal/main/run_main_module.js:17:47[39m
```

1. 예외와 기본 예외 처리

- [예제 8-1] TypeError 기본 예외 처리
  - 1 예외 상황 확인 : undefined 자료형을 일반적인 객체 또는 함수처럼 다루면 TypeError 예외가 발생

코드 8-1 TypeError 발생 — typeError.js

```
// 함수 선언
function callThreeTimes(callback) {
    for (let i = 0; i < 3; i++) {
        callback();
    }
}

// 정상 실행
callThreeTimes(function () { console.log('안녕하세요'); });

// 예외 발생
callThreeTimes();
```

실행 결과

```
안녕하세요
안녕하세요
안녕하세요
C:\example\typeError.js:4
        callback();
        ^
```

함수가 아니므로 호출할 수 없습니다. 따라서 예외가 발생합니다.

```
TypeError: callback is not a function
    at callThreeTimes (C:\Users\KDY\Desktop\test.js:4:9)
    at Object.<anonymous> (C:\Users\KDY\Desktop\test.js:12:1)
    at Module._compile (internal/modules/cjs/loader.js:1085:14)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)
    at Module.load (internal/modules/cjs/loader.js:950:32)
    at Function.Module._load (internal/modules/cjs/loader.js:790:14)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:
76:12)
    at internal/main/run_main_module.js:17:47
```

1. 예외와 기본 예외 처리

- 2 기본 예외 처리 : 사전에 해당 데이터가 undefined인지 조건문으로 확인

코드 8-2 TypeError를 기본 예외 처리로 처리 typeError.js

```
// 함수 선언
function callTenTimes(callback) {
    if (typeof(callback) == "function") {
        for (let i = 0; i < 10; i++) {
            callback();
        }
    } else {
        console.log('매개변수 callback을 함수로 지정해주세요.');
    }
}

// 정상 실행
```

callback이 함수 자료형일 때만 호출합니다.

1. 예외와 기본 예외 처리

- 2 기본 예외 처리 : 사전에 해당 데이터가 undefined인지 조건문으로 확인

```
callTenTimes(function () { console.log('안녕하세요'); });

// 예외 발생
callTenTimes();
```

실행 결과

```
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
안녕하세요
매개변수 callback을 함수로 지정해주세요.
```

2. 고급 예외 처리

- try catch finally 구문

```
try {
    // 예외가 발생하면
} catch (exception) {
    // 여기서 처리합니다.
} finally {
    // 여기는 무조건 실행합니다.
}
```

  - catch 구문, finally 구문 생략 가능

```
try {
    // 예외가 발생하면
} catch (exception) {
    // 여기서 처리합니다.
}
```

```
try {
    // 예외가 발생하면
} finally {
    // 여기는 무조건 실행합니다.
}
```

2. 고급 예외 처리

- [예제 8-2] 고급 예외 처리
  - 1 예외 상황 확인 : 배열을 생성할 때 길이를 음수로 지정하면 RangeError가 발생

코드 8-3 예외 상황 확인 tryCatchBasic.js

```
// 예외를 발생시킵니다.
const array = new Array(-2000);
```

실행 결과

```
C:\example\tryCatchBasic.js:2
const array = new Array(-2000);
              ^


RangeError: Invalid array length
    at Object.<anonymous> (C:\example\tryCatchBasic.js:2:15)
[90m    at Module._compile (internal/modules/cjs/loader.js:1085:14)[39m
[90m    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)[39m
[90m    at Module.load (internal/modules/cjs/loader.js:950:32)[39m
[90m    at Function.Module._load (internal/modules/cjs/loader.js:790:14)[39m
[90m    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:
76:12)[39m
[90m    at internal/main/run_main_module.js:17:47[39m
```

2. 고급 예외 처리

- 2 고급 예외 처리 : try catch finally 구문

코드 8-4 고급 예외 처리 tryCatchBasic.js

```
try {
    // 예외를 발생시킵니다.
    const array = new Array(-2000);
} catch (exception) {
    console.log(`${exception.name} 예외가 발생했습니다.`);
} finally {
    console.log('finally 구문이 실행되었습니다.');
}
```

실행 결과

```
RangeError 예외가 발생했습니다.
finally 구문이 실행되었습니다.
```

3. 예외 객체

- 예외가 발생하면 어떤 예외가 발생했는지 정보를 전달함
- catch 구문의 괄호 안의 변수
- name 속성과 message 속성이 있음

```
try {

} catch (exception) {

}
```

3. 예외 객체

- [예제 8-3] 예외 객체
  - ReferenceError 후에 예외 객체의 name 속성과 message 속성을 출력

코드 8-8 예외 객체 exceptionObject.js

```
try {
    // 예외를 발생시킵니다.
    error.error.error();
} catch (e) {
    console.log(e.name);
    console.log(e.message);
}
```

실행 결과

```
ReferenceError
error is not defined
```

4. 예외 강제 발생

- throw 키워드 사용
- throw 키워드 뒤에는 문자열 또는 Error 객체를 입력

코드 8-9 간단한 예외 강제 발생 throw.js

```
throw '강제 예외';
```

실행 결과

```
C:\example\throw.js:1
throw '강제 예외';
^
강제 예외
(Use `node --trace-uncaught ...` to show where the exception was thrown)
```

4. 예외 강제 발생

- 자세한 예외 출력은 Error 객체를 사용
  어떤 파일의 몇 번째 줄에서 예외가 발생했는지도 확인가능

코드 8-10 Error 객체를 사용한 예외 강제 발생 throw.js

```
// 예외 객체를 만듭니다.
error.name = '내 마음대로 오류';
error.message = '오류의 메시지';

// 예외를 발생시킵니다.
throw error;
```

실행 결과

```
C:\example\throw.js:9
throw error;
^


내 마음대로 오류: 오류의 메시지
    at Object.<anonymous> (C:\Users\KDY\Desktop\test.js:2:15)
[90m    at Module._compile (internal/modules/cjs/loader.js:1085:14)[39m
[90m    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1114:10)[39m
[90m    at Module.load (internal/modules/cjs/loader.js:950:32)[39m
[90m    at Function.Module._load (internal/modules/cjs/loader.js:790:14)[39m
[90m    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:
76:12)[39m
[90m    at internal/main/run_main_module.js:17:47[39m
```

4. 예외 강제 발생

- 문자열을 사용할 때는 catch 구문의 예외 객체에 간단한 문자열만 전달됨

코드 8-11 간단한 예외 강제 발생 때의 예외 객체 throw.js

```
try {
    // 예외를 강제로 발생시킵니다.
    throw '예외가 발생했습니다';
} catch (exception) {
    // 예외 객체를 출력합니다.
    console.log('예외가 발생했습니다.');
    console.log(exception);
}
```

실행 결과

```
예외가 발생했습니다.
예외가 발생했습니다
```

4. 예외 강제 발생

- Error 객체를 사용한 예외 강제 발생 때의 예외 객체

코드 8-12 Error 객체를 사용한 예외 강제 발생 때의 예외 객체 throw.js

```
try {
    // 예외 객체를 만듭니다.
    const error = new Error('메시지');
    error.name = '내 마음대로 오류';
    error.message = '오류의 메시지';

    // 예외를 발생시킵니다.
    throw error;
} catch (exception) {
    // 예외 객체를 출력합니다.
    console.log(`${exception.name} 예외가 발생했습니다.`);
    console.log(exception.message);
}
```

실행 결과

```
내 마음대로 오류 예외가 발생했습니다.
오류의 메시지
```


THANK
YOU!