
모던 자바스크립트

Deep Dive

47장_에러 처리

에러 처리의 필요성

에러가 발생하지 않는 코드를 작성하는 것은 불가능
발생한 에러를 방치하면 프로그램 강제 종료
이 경우 원인을 파악하여 대응하기 어려움

언제나 **에러나 예외적인 상황**(에러를 발생하지 않는 상황)이 발생할 수 있다는 것을 전제하고 대응하는 코드를 작성하는 것이 중요

※ 예외적인 상황은 에러로 이어질 가능성이 큼

```
1 console.log('[Start]');
2
3 // 에러를 방치하면 프로그램은 종료
4 foo();
5
6 console.log('[End]');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] node "c:\Users\kyh\Desktop\coding\Algo\note.js"
[Start]
c:\Users\kyh\Desktop\coding\Algo\note.js:4
foo();
^
```

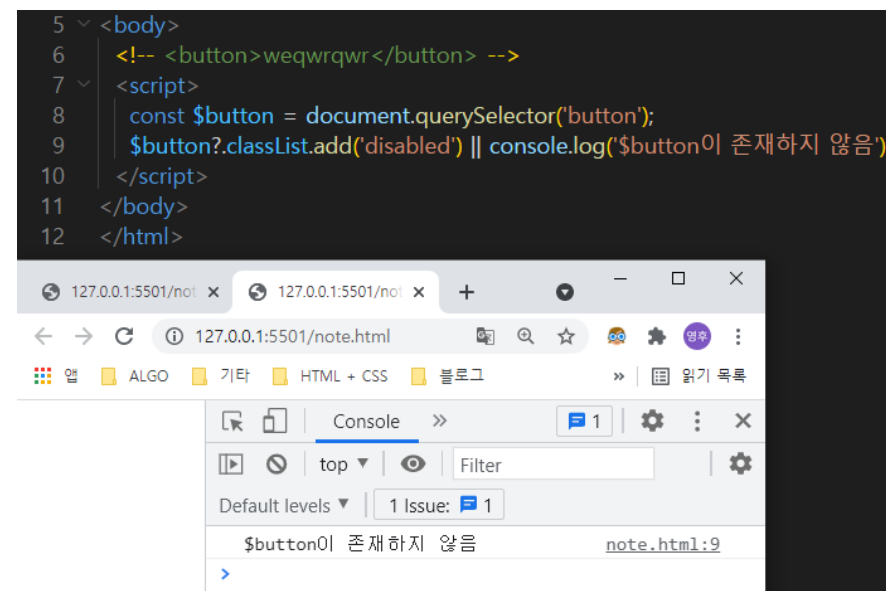
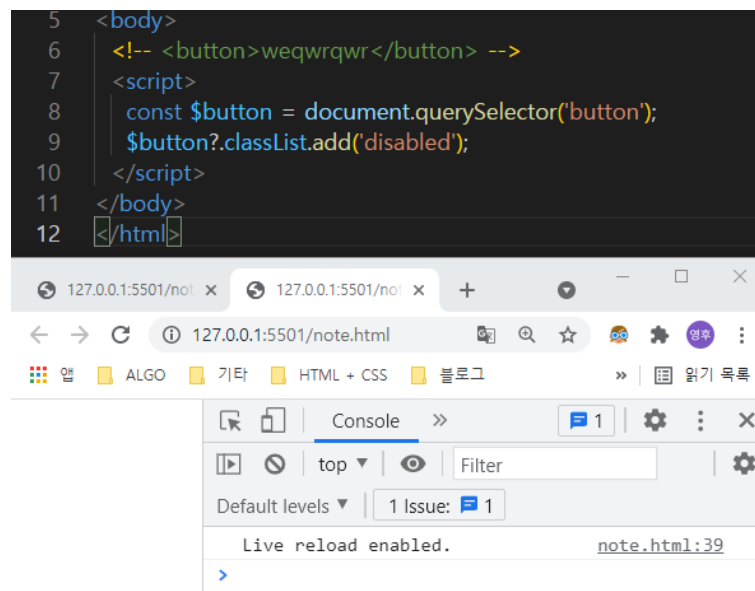
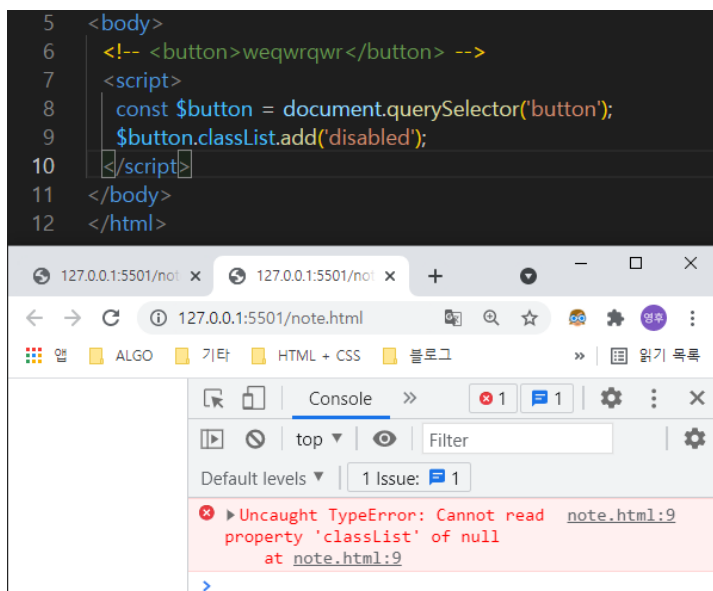
```
1 // CSS 선택자 문법에 맞지 않는 경우
2 const $elem = document.querySelector('#1');
3 // DOM Exception : Failed to execute 'querySelector' on 'Document' : '#1' is not a valid selector
```

```
1 // 노드가 존재하지 않는 경우
2 const $button = document.querySelector('button');
3 $button.classList.add('disabled');
4 // TypeError : Cannot read Property 'classList' of null
```

에러 처리 방법

(1) 단축 평가 혹은 옵셔널 체이닝

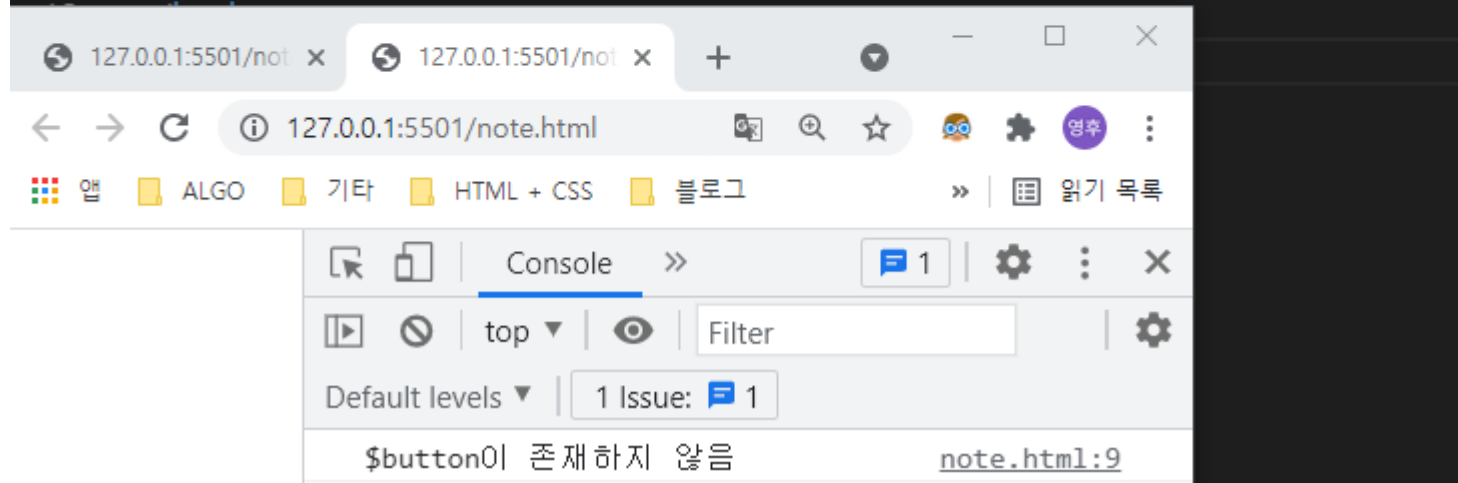
```
1 // 노드가 존재하지 않는 경우
2 const $button = document.querySelector('button');
3 $button.classList.add('disabled');
4 // TypeError : Cannot read Property 'classList' of null
```



에러 처리의 필요성

if 문에서도 작성이 가능

```
5 <body>
6 <!-- <button>weqwrqwr</button> -->
7 <script>
8   const $button = document.querySelector('button');
9   if($button?.classList.add('disabled') || console.log('$button이 존재하지 않음')) {
10     console.log('hi');
11   }
12 </script>
```



try... catch... finally 문

(2) try... catch... finally 문

일반적으로 이 방법을 에러 처리(Error Handling)라고 함

finally, catch 문은 생략 가능. catch문이 없는 try문은 의미가 없음

try... catch... finally 문으로 에러를 처리하면 **프로그램이 강제 종료되지 않음**

```
try {  
    // 실행할 코드(에러가 발생할 가능성이 있는 코드)  
} catch(err) {  
    // try 코드 블록에서 에러가 발생하면 이 코드 블록의 코드가 실행  
    // err에는 try 코드 블록에서 발생한 Error 객체가 전달  
    // catch 코드 블록에서만 유효  
} finally {  
    // 에러 발생과 상관없이 반드시 한 번 실행됨  
}
```

Error 객체

Error 생성자 함수는 **에러 객체**를 생성

Error 생성자 함수에는 **에러를 설명하는 에러 메시지**를 인수로 전달

```
const error = new Error('invalid');
```

에러 객체를 생성할 수 있는 **7가지 Error 생성자 함수**

생성된 인스턴스는 Error.prototype을 상속받음

생성자 함수	인스턴스
Error	일반적인 에러 객체
SyntaxError	문법에 맞지 않는 문 을 해석할 때 발생하는 에러 객체
ReferenceError	참조할 수 없는 식별자 를 참조했을 때 발생하는 에러 객체
TypeError	피연산자 또는 인수의 데이터 타입 이 유효하지 않을 때 발생하는 에러 객체
RangeError	숫자값의 허용 범위 를 벗어났을 때 발생하는 에러 객체
URIError	encodeURIComponent 혹은 decodeURI 함수에 부적절한 인수 를 전달했을 때 발생하는 에러 객체
EvalError	Eval 함수 에서 발생하는 에러 객체

Error 객체

```
1@1; // SyntaxError : Invalid or unexpected Token  
foo(); // ReferenceError : foo is not defined  
null.foo; // TypeError : Cannot read property 'foo' of null  
new Array(-1); // RangeError : Invalid array length;  
decodeURIComponent('%'); // wURIError: URI malformed
```


throw 문

에러 객체 생성과 에러 발생은 의미가 다름

에러 발생은 **try** 코드 블록에서 **throw**문으로 에러 객체를 던짐

throw 표현식;

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

에러를 발생시킨 콜 스택의 호출 정보

에러 메시지

```
1 const repeat = (n, f) => {  
2   // 매개변수 f에 전달된 인수가 함수가 아니면 TypeError  
3   if(typeof f !== 'function') throw new TypeError('f must be a function');  
4  
5   for(var i = 0; i < n; i++) {  
6     f(i)  
7   }  
8 }  
9  
10 try {  
11   repeat(2, 1);  
12 } catch(err) {  
13   console.log(Object.getOwnPropertyDescriptors(err));  
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
{  
  stack: {  
    value: 'TypeError: f must be a function\n' +  
    '    at repeat (c:\\Users\\WWkyh\\Desktop\\coding\\Algo\\note.js:3:37)\n' +  
    '    at Object.<anonymous> (c:\\Users\\WWkyh\\Desktop\\coding\\Algo\\note.js:11:3)\n' +  
    '    at Module._compile (internal/modules/cjs/loader.js:1063:30)\n' +  
    '    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1092:10)\n' +  
    '    at Module.load (internal/modules/cjs/loader.js:928:32)\n' +  
    '    at Function.Module._load (internal/modules/cjs/loader.js:769:14)\n' +  
    '    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:72:12)\n' +  
    '    at internal/main/run_main_module.js:17:47',  
    writable: true,  
    enumerable: false,  
    configurable: true  
  },  
  message: {  
    value: 'f must be a function',  
    writable: true,  
    enumerable: false,  
    configurable: true  
  }  
}
```

에러의 전파

```
1  const foo = () => {
2    throw Error('foo에서 발생한 에러');
3  }
4
5  const bar = () => {
6    try {
7      foo();
8    } catch(err) {
9      console.log(Object.getOwnPropertyDescriptors(err));
10     }
11  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Running] node "c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js"

stack: {
 value: 'Error: foo에서 발생한 에러\n' +
 ' at foo (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:2:9)\n' +
 ' at bar (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:7:5)\n' +
 ' at baz (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:14:3)\n' +
 ' at Object.<anonymous> (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:18:3)\n' +
 ' at Module._compile (internal/modules/cjs/loader.js:1063:30)\n' +
 ' at Object.Module._extensions..js (internal/modules/cjs/loader.js:1092:10)\n' +
 ' at Module.load (internal/modules/cjs/loader.js:928:32)\n' +
 ' at Function.Module._load (internal/modules/cjs/loader.js:769:14)\n' +
 ' at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:72:12)\n' +
 ' at internal/main/run_main_module.js:17:47',
 writable: true,
 enumerable: false,
 configurable: true
},
message: {
 value: 'foo에서 발생한 에러',
 writable: true,
 enumerable: false,
 configurable: true
}

catch

```
1  const foo = () => {
2    throw Error('foo에서 발생한 에러');
3  }
4
5  const bar = () => {
6    try {
7      foo();
8    } catch(err) {
9      console.log(err);
10     }
11  };
12
13  const baz = () => {
14    bar();
15  }
16
17  try {
18    baz();
19  } catch(err) {
20    console.error(err);
21  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Running] node "c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js"

Error: foo에서 발생한 에러
 at foo (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:2:9)
 at bar (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:7:5)
 at baz (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:14:3)
 at Object.<anonymous> (c:\Users\Wkyh\Desktop\coding\WAlgo\Wnote.js:18:3)
 at Module._compile (internal/modules/cjs/loader.js:1063:30)
 at Object.Module._extensions..js (internal/modules/cjs/loader.js:1092:10)
 at Module.load (internal/modules/cjs/loader.js:928:32)
 at Function.Module._load (internal/modules/cjs/loader.js:769:14)
 at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:72:12)
 at internal/main/run_main_module.js:17:47

Done] exited with code=0 in 0.164 seconds

에러는 호출자 방향으로 전파

throw 문

throw된 에러를 캐치하여 적절히 대응하면, 프로그램을 강제 종료시키지 않고 코드 실행 흐름 복구 가능
throw된 에러를 어디에서도 캐치하지 않으면 프로그램은 강제 종료

주의할 것

비동기 함수인 **setTimeout**, 프로미스 후속 처리 메서드의 콜백 함수는 호출자가 없음
콜 스택의 가장 하부에 존재하게 되므로, 에러를 전파할 호출자가 존재하지 않음