모던 자바스크립트
Deep Dive

20장_strict모드

strict 모드란?

```
function foo() {
        x = 10;
}

foo();
console.log(x); ?
```

선언이 존재하지 않기 때문에, ReferenceError를 발생 X
window 객체의 window 프로퍼티에 할당(**암묵적 전역**) → 오류를 발생시킬 가능성이 높으므로 반드시 키워드 사용

하지만 오타나 문법 지식의 미비로 인한 실수는 언제나 발생
근본적인 해결책은 오류를 발생시키기 어려운 환경에서 개발하는 것 → ES5부터 **strict mode** 등장(엄격 모드)

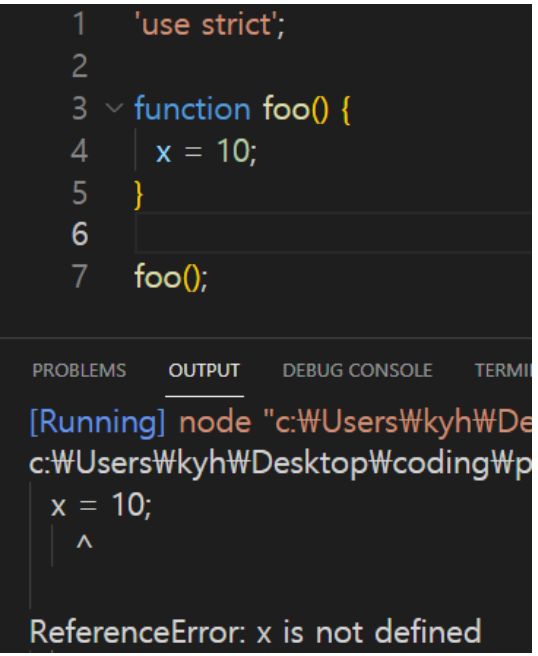
strict mode란 자바스크립트 언어의 문법을 좀 더 엄격히 적용하여, 오류를 발생시킬 가능성이 높거나, 자바스크립트 엔진의 최적화 작업에 문제를 일으킬 수 있는 코드에 대해 명시적인 에러를 발생시킴

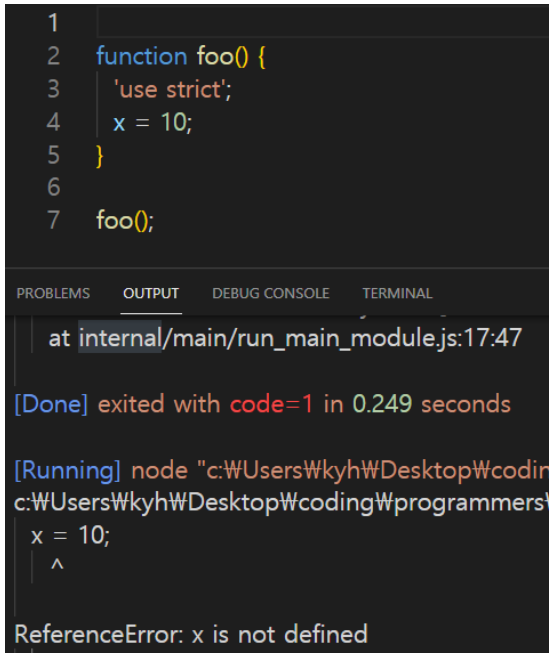
ESLint 같은 린트 도구를 사용해도 strict mode와 유사한 효과를 얻을 수 있음.
ESLint는 문법적 오류만이 아니라 잠재적 오류까지 찾아내고, 코딩 컨벤션을 설정 파일 형태로 강제할 수 있기 때문에 더욱 강력한 효과를 얻을 수 있음. 필자는 strict mode보다 린트 도구의 사용을 선호

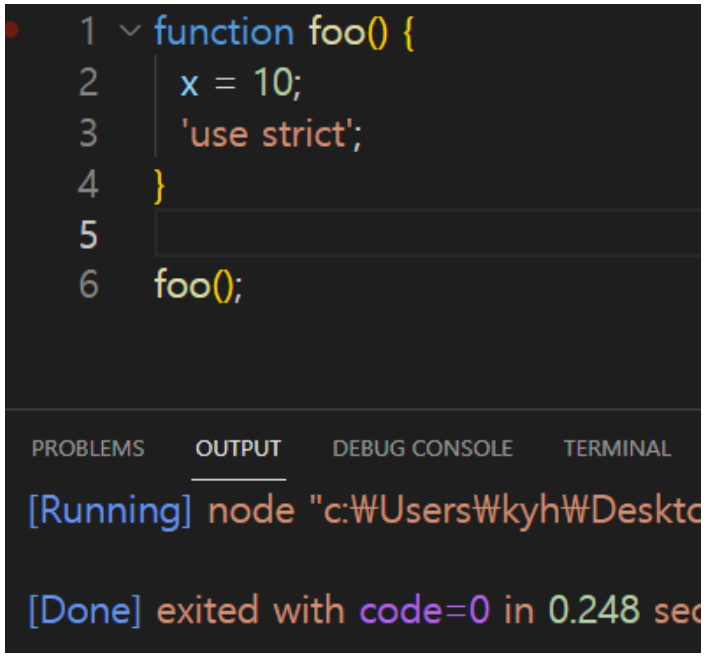
ES6에서 도입된 클래스와 모듈은 기본적으로 strict mode가 적용됨

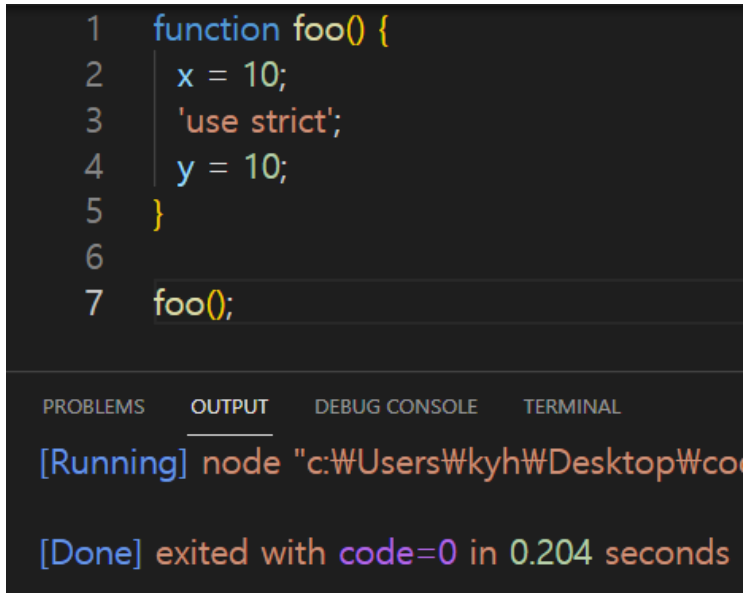
strict 모드 적용

- 전역의 선두 또는 함수 몸체의 선두에 'use strict' 추가
- 전역의 선두에 추가하면 스크립트 전체에 strict mode 적용
- 함수 몸체의 선두에 추가하면 해당 함수와 중첩 함수에 strict mode가 적용
- 코드의 선두에 'use strict'를 위치시키지 않으면 strict mode가 동작하지 않음









전역에 strict mode를 적용하는 것은 피하자

- 전역에 적용한 strict mode는 스크립트 단위로 적용

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head> </head>
4      <body>
5          <script>
6              'use strict';
7          </script>
8          <script>
9              x = 1;
10             console.log(x); // 1, 에러가 발생하지 않음
11         </script>
12         <script>
13             'use strict';
14
15             y = 1; // Reference Error
16             console.log(y);
17         </script>
18     </body>
19 </html>
```

전역에 strict mode를 적용하는 것은 피하자

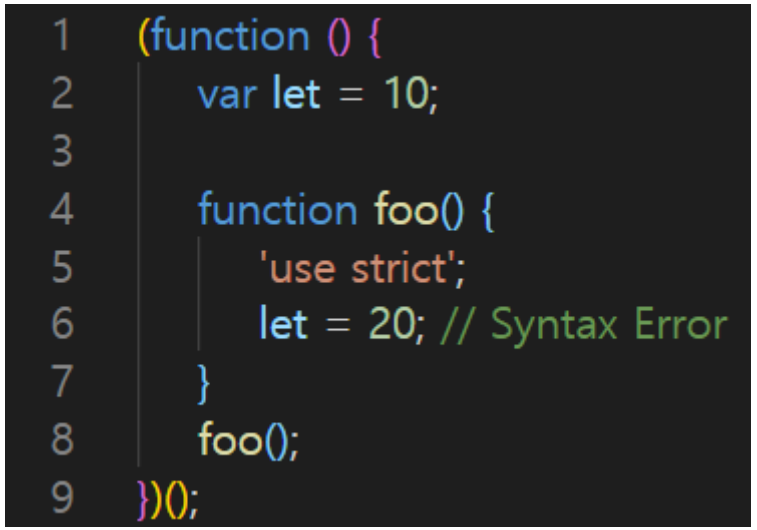
- strict mode 스크립트와 non-strict mode 스크립트를 혼용하는 것은 오류 발생 가능성 존재
특히 외부 서드파티 라이브러리를 사용하는 경우 라이브러리가 non-strict mode인 경우도 있기 때문에 전역에 strict mode를 적용하는 것은 바람직하지 않음

이 경우, 즉시 실행 함수로 스크립트 전체를 감싸서 스코프를 구분하고, 즉시 실행 함수의 선두에 strict mode를 적용한다.

```
1  (function() {
2      'use strict';
3      // Do something
4  })();
```

함수 단위로 strict mode를 적용하는 것도 피하자

- 어떤 함수는 strict mode, 어떤 함수는 non-strict mode는 바람직하지 않음
- 모든 함수에 strict mode 적용은 번거로움
- strict mode가 적용된 함수가 참조할 함수 외부 컨텍스트에 strict mode 적용도 필수



```
1  (function () {
2      var let = 10;
3
4      function foo() {
5          'use strict';
6          let = 20; // Syntax Error
7      }
8      foo();
9  })();
```

- **따라서 strict mode는 즉시 실행 함수로 감싼 스크립트 단위로 적용하는 것이 바람직**

strict mode가 발생시키는 에러

1. 암묵적 전역
2. 변수, 함수, 매개변수의 삭제
delete 연산자로 변수, 함수, 매개변수를 삭제하면 SyntaxError가 발생

```
1  (function () {
2      'use strict';
3
4      var x = 1;
5      delete x; // SyntaxError : delete of an unqualified identifier in strict mode
6
7      function foo(a) {
8          delete(a); // SyntaxError : delete of an unqualified identifier in strict mode
9      }
10     delete foo; // SyntaxError : delete of an unqualified identifier in strict mode
11 })();
```

3. 매개변수 이름의 중복

```
1  (function () {
2      'use strict';
3
4      function foo(x, x) { // SyntaxError : Duplicate parameter name not allowed in this context
5          return x + x;
6      }
7
8      console.log(foo(1,2));
9  })();
```

strict mode가 발생시키는 에러

4. width 문의 사용
- 전달된 객체를 스코프 체인에 추가
- 객체의 프로퍼티를 반복해서 사용할 때 객체 이름을 생략할 수 있어서 코드가 간단해지는 효과
- **가독성과 성능이 나빠짐. 사용하지 않는 것이 좋음**

```
1   (function () {
2       console.log(x); // x is not defined
3       console.log(y); // y is not defined
4       console.log(z); // undefined
5       with({x : 1}) {
6           console.log(x); // 1
7           let y = 2;
8           var z = 3;
9           console.log(y) // 2
10      }
11  })()
```

```
1   (function () {
2       'use strict';
3
4       with({x : 1}) {
5           console.log(x) // 1
6       }
7   })()
8
9
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] node "c:₩Users₩kyh₩Desktop₩coding₩programmers₩
c:₩Users₩kyh₩Desktop₩coding₩programmers₩개인 공부자료(메
    with({x : 1}) {
    ^^^^

SyntaxError: Strict mode code may not include a with statement
```

strict mode 적용에 의한 변화

1. 일반 함수의 this

- 일반 함수로서 호출하면 this에 undefined
- 생성자 함수가 아닌 일반 함수 내에서는 this 사용 안하기때문

```
(function() {
    function foo() {
        console.log(this);
    }
    foo();

    function Foo() {
        console.log(this);
    }
    new Foo();
})();
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:₩Users₩kyh₩Desktop₩coding₩programmers₩7
<ref *1> Object [global] {
  global: [Circular *1],
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  },
  queueMicrotask: [Function: queueMicrotask],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  }
}
Foo {}
```

```
(function() {
    'use strict';

    function foo() {
        console.log(this); // undefined
    }
    foo();

    function Foo() {
        console.log(this); // Foo{}
    }
    new Foo();
})();
```

strict mode 적용에 의한 변화

2. arguments 객체
- 매개변수에 전달된 인수를 재할당하여 변경해도 arguments 객체에 반영안됨