

ECMAScript 6

변수와 식별자

변수와 식별자

식별자 정의와 특징

- 식별자(identifier)는 변수를 구분할 수 있는 변수명을 말함
- 식별자는 반드시 문자, 달러(\$) 또는 밑줄(_)로 시작
- 대소문자를 구분하며, 클래스명 외에는 모두 소문자로 시작
- 예약어* 사용 불가능
 - 예약어 예시: for, if, case 등

식별자 작성 스타일

- 카멜 케이스(camelCase, lower-camel-case)
 - 변수, 객체, 함수에 사용
- 파스칼 케이스(PascalCase, upper-camel-case)
 - 클래스, 생성자에 사용
- 대문자 스네이크 케이스(SNAKE_CASE)
 - 상수(constants)*에 사용
 - 상수의 정의: 개발자의 의도와 상관없이 변경될 가능성이 없는 값을 말함

식별자 작성 스타일

```
1 // 변수
2 let dog
3 let variableName
4
5 // 객체
6 const userInfo = { name: 'juan', age: 27 }
7
8 // 함수
9 function getPropertyName () {}
10 function onClick () {}
```

카멜 케이스 (camelCase)

두 번째 단어의 첫 글자부터 대문자

식별자 작성 스타일

```
1 // 클래스
2 class User {
3     constructor(options) {
4         this.name = options.name
5     }
6 }
7
8 // 생성자
9 const good = new User({
10     name: '홍길동',
11 })
```

파스칼 케이스 (PascalCase)

모든 단어의 첫 번째 글자를 대문자로 작성

식별자 작성 스타일

```
1 // 상수
2 const API_KEY = 'SOMEKEY'
3 const PI = Math.PI
```

```
// 상수가 아닌 경우
const mutableCollection = new Set()
```

대문자 스네이크 케이스 (SNAKE_CASE)

모든 단어 대문자 작성 & 단어 사이에 언더스코어 삽입

변수 선언 키워드 (let, const)

const

- 재할당할 수 없는 변수 선언 시 사용
- 변수 재선언 불가능
- 블록 스코프*

let

- 재할당 할 수 있는 변수 선언 시 사용
- 변수 재선언 불가능
- 블록 스코프*

(참고) 선언, 할당, 초기화

```
let foo          // 선언
console.log(foo) // undefined

foo = 11         // 할당
console.log(foo) // 11

let bar = 0      // 선언 + 할당
console.log(bar) // 0
```

- 선언 (Declaration)
 - 변수를 생성하는 행위 또는 시점
- 할당 (Assignment)
 - 선언된 변수에 값을 저장하는 행위 또는 시점
- 초기화 (Initialization)
 - 선언된 변수에 처음으로 값을 저장하는 행위 또는 시점

변수 선언 키워드 (let, const) 예시 (1) - 재할당

let (재할당 가능)

```
let number = 10    // 1. 선언 및 초기값 할당  
number = 10        // 2. 재할당  
  
console.log(number) // 10
```

const (재할당 불가능)

```
const number = 10  // 1. 선언 및 초기값 할당  
number = 10        // 2. 재할당 불가능  
  
=> Uncaught TypeError  
:Assignment to constant variable.
```

변수 선언 키워드 (let, const) 예시 (2) - 재선언

let (재선언 불가능)

```
let number = 10 // 1. 선언 및 초기값 할당  
let number = 50 // 2. 재선언 불가능  
  
=> Uncaught SyntaxError  
      : Identifier 'number' has already been declared
```

const (재선언 불가능)

```
const number = 10 // 1. 선언 및 초기값 할당  
const number = 50 // 2. 재선언 불가능  
  
=> Uncaught SyntaxError  
      : Identifier 'number' has already been declared
```

변수 선언 키워드 (let, const)

```
let x = 1

if (x === 1) {
  let x = 2
  console.log(x) // 2
}

console.log(x) // 1
```

- 블록 스코프* (block scope)
- if, for, 함수 등의 **중괄호 내부**를 가리킴
- 블록 스코프를 가지는 변수는
블록 바깥에서 접근 불가능

변수와 식별자 실습 (1)

- 목표: 블록 스코프의 이해 및 let, const 키워드 연습 (01-variables.js)
- 문제: 파일에 작성된 주석 참고

변수 선언 키워드 (var)

- var
 - var로 선언한 변수는 재선언 및 재할당 모두 가능
 - ES6 이전에 변수를 선언할 때 사용되던 키워드
 - 호이스팅*되는 특성으로 인해 예기치 못한 문제 발생 가능
 - 따라서 ES6 이후부터는 var 대신 const와 let을 사용하는 것을 권장
- 함수 스코프*

변수 선언 키워드 (var)

```
var number = 10 // 1. 선언 및 초기값 할당  
var number = 50 // 2. 재할당  
  
console.log(number) // 50
```

재선언 및 재할당 모두 가능

변수 선언 키워드 (var)

```
1 function foo() {  
2   var x = 5  
3   console.log(x)  // 5  
4 }  
5  
6 // ReferenceError: x is not defined  
7 console.log(x)
```

- 함수 스코프* (function scope)
 - 함수의 중괄호 내부를 가리킴
 - 함수 스코프를 가지는 변수는 함수 바깥에서 접근 불가능

변수 선언 키워드 (var)

```
console.log(username) // undefined
var username = '홍길동'

console.log(email)      // Uncaught ReferenceError
let email = 'gildong@gmail.com'

console.log(age)        // Uncaught ReferenceError
const age = 50
```

- 호이스팅* (hoisting)
 - 변수를 선언 이전에 참조할 수 있는 현상
 - 변수 선언 이전의 위치에서 접근 시 undefined를 반환

변수와 식별자 실습 (2)

- 목표: 함수 스코프, 호이스팅의 이해 및 var 키워드 연습 (01-variables.js)
- 문제: 파일에 작성된 주석 참고

변수와 식별자 Quiz

Q1. 자바스크립트 변수 선언 시 사용 가능한 키워드는 const와 let 뿐이다. T/F

Q2. let 키워드로 선언한 변수는 재할당이 가능하다. T/F

Q3. const 키워드로 선언한 변수는 재할당이 가능하다. T/F

변수와 식별자 Quiz

Q1. 자바스크립트 변수 선언 시 사용 가능한 키워드는 const와 let 뿐이다.

F

A1. 자바스크립트 변수 사용 시 사용 가능한 키워드는 const, let 그리고 var이다.

Q2. const 키워드로 선언한 변수는 재할당이 가능하다.

F

A2. const 키워드로 선언한 변수는 재할당이 불가능하다.

Q3. let 키워드로 선언한 변수는 재할당이 가능하다.

T

let, const, var 비교

키워드	재선언	재할당	스코프	비고
let	X	O	블록 스코프	ES6부터 도입
const	X	X	블록 스코프	ES6부터 도입
var	O	O	함수 스코프	사용 X