

15장

let, const 키워드와 블록 레벨 스코프



모던 자바스크립트 Deep Dive

목차

1. var 키워드

2. let 키워드

3. const 키워드

1. var 키워드로 선언한 변수의 문제점

변수 중복 선언 허용

- var 키워드로 선언된 변수는 같은 스코프 내에서 **중복 선언을 허용함**

```
var x = 1;
```

```
var y = 1;
```

```
var x = 100;
```

→ var 키워드가 없는 것처럼 동작

```
var y;
```

→ 무시됨

```
console.log(x); // 100
```

```
console.log(y); // 1
```

1. var 키워드로 선언한 변수의 문제점

함수 레벨 스코프

- var 키워드로 선언한 변수는 함수의 코드 블록만을 지역 스코프로 인정함

```
var i = 10;           전역 변수  
  
for (var i = 0; i < 5; i++) {  
    console.log(i); // 0 1 2 3 4  
}  
  
console.log(i); // 5
```

- 전역 변수가 중복 선언될 가능성

1. var 키워드로 선언한 변수의 문제점

변수 호이스팅

- var 키워드로 선언한 변수는 변수 호이스팅에 의해 변수 선언문 이전에 참조 가능

→ 1. foo 변수 선언 / 2. undeifined로 초기화

```
console.log(foo); // undefined
```

foo = 123; → 3. 변수에 값 할당

```
console.log(foo); // 123
```

```
var foo;
```

2. let 키워드

변수 중복 선언 금지

- var 키워드와 다르게, let 키워드로 변수를 중복 선언하면 문법 에러 발생

```
let bar = 123;
```

```
let bar = 456;
```

```
// SyntaxError: Identifier 'bar' has already been  
declared
```

2. let 키워드

블록 레벨 스코프

- let 키워드로 선언한 변수는 모든 코드 블록 (함수, if문, for문, try/catch문 등)을 지역 스코프로 인정

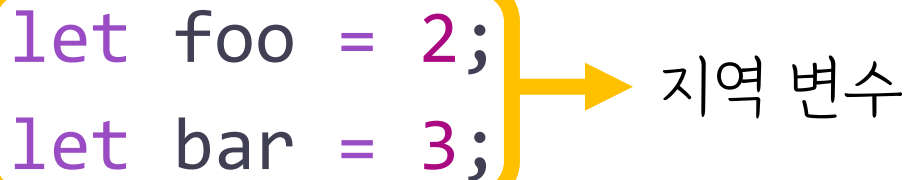
```
let foo = 1;
```



전역 변수

```
{
```

```
    let foo = 2;  
    let bar = 3;
```



지역 변수

```
}
```

```
console.log(foo); // 1
```

```
console.log(bar); // ReferenceError: bar is not defined
```

2. let 키워드

변수 호이스팅

- let 키워드로 선언한 변수는 변수 호이스팅이 발생하지 않는 것처럼 동작함

→ 선언 단계 실행

```
console.log(foo); // ReferenceError: foo is not defined
```

let foo; → 변수 선언문에서 초기화 단계 실행

```
console.log(foo); // undefined
```

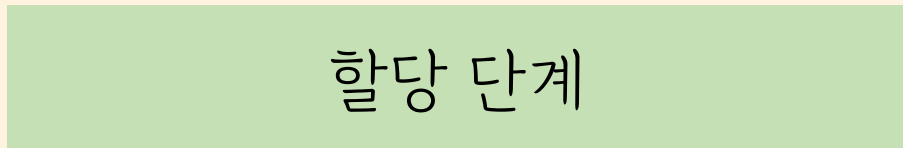
foo = 1; → 할당문에서 할당 단계 실행

```
console.log(foo); // 1
```


2. let 키워드

변수 호이스팅

- let 키워드로 선언한 변수는 변수 호이스팅이 발생하지 않는 것처럼 동작함



var 키워드로 선언한 변수



let 키워드로 선언한 변수

2. let 키워드

변수 호이스팅

- let 키워드로 선언한 변수도 호이스팅이 발생하지 않는 것은 아님

```
let foo = 1; → 전역 변수
{
  console.log(foo); // ReferenceError
  let foo = 2; → 지역 변수
}
```

2. let 키워드

전역 객체와 let

- var 키워드로 선언한 전역 변수, 전역 함수 등은 전역 객체 window의 프로퍼티
- let 키워드로 선언한 전역 변수는 전역 객체의 프로퍼티 x
 - > `window.foo` 와 같이 접근 x

3. const 키워드

선언과 초기화

- const 키워드로 선언한 변수는 반드시 선언과 동시에 초기화해야 함

```
const foo;
```

```
// SyntaxError: Missing initializer in const declaration
```

- const 키워드로 선언한 변수는 블록 레벨 스코프를 가짐
- const 키워드로 선언한 변수는 변수 호이스팅이 발생하지 않는 것처럼 동작함

3. const 키워드

재할당 금지 / 상수

- const 키워드로 선언한 변수는 재할당이 금지됨

```
const foo = 1;  
foo = 2; // TypeError: Assignment to constant variable.
```

- const 키워드를 상수를 표현하는 데 사용하기도 함
- 상태 유지, 가독성, 유지보수성 향상

3. const 키워드

const 키워드와 객체

- const 키워드로 선언된 변수에 객체를 할당할 경우에는 값 변경 가능

```
const person = {  
  name: 'Lee'  
};
```

```
person.name = 'Kim';
```

```
console.log(person); // {name: "Kim"}
```

- const 키워드가 불변을 의미하는 것은 아님

var vs. let vs. const

- 변수 선언에는 기본적으로 const 사용
- 재할당이 필요한 경우에 한정해서 let 사용
- var 는 사용하지 않는 것이 좋음