let, var, const

var, let, const

var는 function scope를 지원하고 let, const는 block scope를 지원한다. 그렇기 때문에 다음과 같이 block level에 foo를 456으로 재선언하는 경우 foo를 456으로 인식하지만, let이나 const는 전역 변수를 읽는다.

```
var foo = 123;

console.log(foo); // 123

{
  var foo = 456;
}

console.log(foo); // 456
```

차이점

var의 문제점은 크게 3가지 이다.

1. 변수 중복 선언이 가능하며, 예기치 못한 값을 반환할 수 있다.
2. funcion scope로 인해 함수 외부에서 선언한 변수는 모두 전역 변수가 된다.
3. 변수 선언문 이전에 변수를 참조하면 언제나 undefined를 반환한다.

let, const는 위의 3가지 문제점을 해결했다.

1. 변수 중복 선언 불가

a. let

- 중복 선언은 불가능하지만 재할당은 가능하다.

```
let name = 'yjh'
console.log(name) // yjh

let name = 'bowdy' // Error

name = 'howdy'
console.log(name) // howdy
```

b. const

- 반드시 선언과 초기화를 동시에 진행해야 한다.
- 재선언이 불가하며, 재할당도 불가능하다.
- 단, 객체는 재할당이 가능하다.

```
const name; // Error
const name = 'yjh'

const name = 'yjh'
name = 'howdy' // Error

const name = {
  eng: 'yjh'
}
name.eng = 'howdy'

console.log(name) // { eng: 'howdy' }
```

c. Block 레벨 스코프

- 코드 블록(ex. 함수, if, for, while, try/catch 등)을 지역 스코프로 인정하는 블록 레벨 스코프를 따른다.

```
let a = 1

if(true) {
  let a = 5
}

console.log(a) // 1

// const도 동일하게 작동한다.
```