스코프

모든 식별자(변수명, 함수명, 클래스 등)는 자신이 선언된 위치에 다른코드가 식별자 자신을 참조 할수있는 유효범위 가 결정된다.

① 함수 스코프. (function level scope) Var

함수 외부에서 함수내부의 변수에 접근불가.

- 전역스코프

```
1  var var1 = 1;
2
3  {
4    var var2 = 2;
5  }
6
7  if (true) {
8    var var3 = 3;
9    if (true) {
10     var var4 = 4;
11   }
12 }
```

- 지역 스코프

```
14 function foo() {
15   var var5 = 5;
16
17   function bar() {
18     var var6 = 6;
19   }
20 }
```

② 블록 스코프 (Block level scope) let/const

블록 외부에서 블록 내부의 변수에 접근 불가.

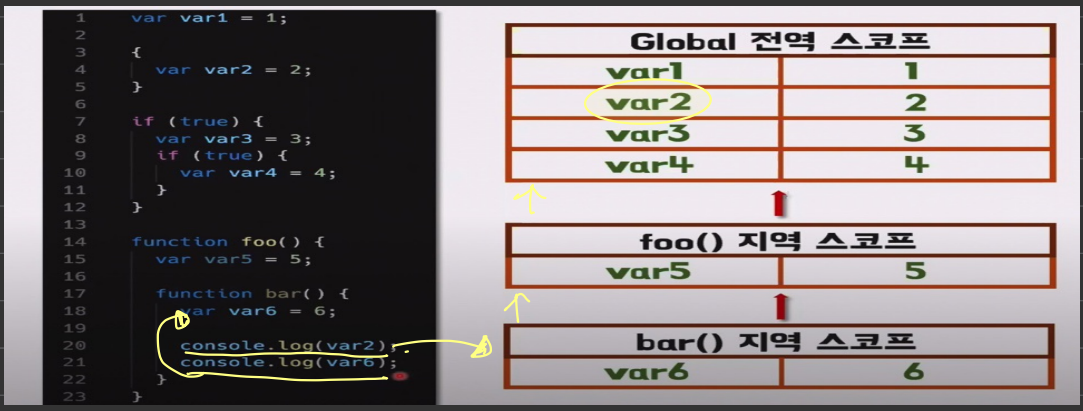
- 전역스코프

```
1  var var1 = 1;
```

- 지역스코프

```
3  {
4    var var2 = 2;
5  }
7  if (true) {
8    var var3 = 3;
9    if (true) {
10     var var4 = 4;
11   }
12 }
14 function foo() {
15   var var5 = 5;
16
17   function bar() {
18     var var6 = 6;
19   }
20 }
```

스코프체인



렉시컬 스코프

```
1  let apple = "apple";
2
3  function fruit(eat) {
4    let apple = "banana";
5    eat();
6  }
7
8  function eat() {
9    console.log(apple);
10 }
11
12 fruit(eat); // ?
13 eat(); // ?
```

eat()의 상위 스코프는?

아래에 따라 상위 스코프가 달라짐

① 함수가 호출된 위치 → 동적스코프

② 함수가 만들어진 위치 → 정적스코프 (Lexical)

banana / apple

apple/apple

=>대다수의 언어가 정적스코프를 따름

클로저 "함수"

상위 스코프의 변수
변수를 가르키는 함수
클로저가 만들어진 환경을 기억.

상위 스코프의 변수를 가르키는 함수
⇒ 클로저가 만들어진 환경을 기억

```
1   function getClosure() {
2     const apple = "사과";
3     return function closure() {
4       return apple;
5     };
6   }
7
8   const closure = getClosure();
9   console.log(closure());
```

출력 값을 예측해보자.

상위 스코프부터 따져보자.

getClosure() 지역 스코프	
apple	"사과"

↑

closure() 지역 스코프

closure() 가 호출되는 위치 : 전역 → Why getClosure의 스코프를 참조?

⇒ 렉시컬 과 정적 스코프 때문에.

```
1   function User() {
2     let _id = "우리밋";
3
4     const closures = {
5       // closure 1
6       getId: function () {
7         return _id;
8       },
9       // closure 2
10      setId: function (id) {
11        _id = id;
12      },
13    };
14
15    return closures;
16  }
```

```
18  const user = new User();
19
20  console.log(user.getId());
21  console.log(user._id);
22
23  user.setId("woorimIT");
24  console.log(user.getId());
25  console.log(user._id);
```

출력 값을 예측해보자.

```
node closure.js
우 리 밋
undefined
woorimIT
undefined
```

클로저를 통해 자바의 Private 를 가능하게 해줌.