JavaScript – Scope, Closure

Sung-Dong Kim,
School of Computer Engineering,
Hansung University

Contents

- Scope (영역)
- Closure 1
- Closure 2
- Closure 3

Scope

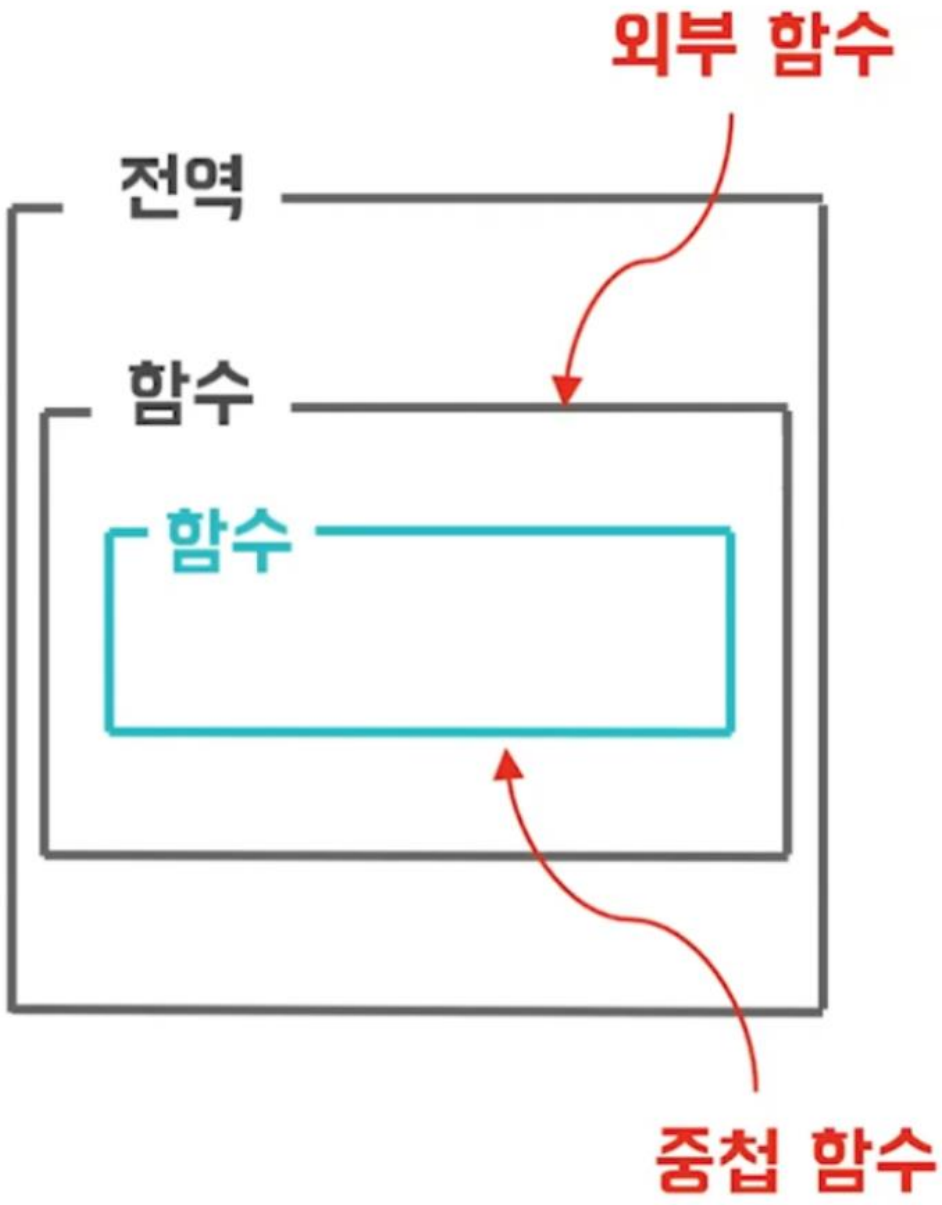
- what: 식별자를 참조할 수 있는 코드의 부분
- var: 함수 레벨 스코프
- let, const: 블록 레벨 스코프 (if, for, 함수, ...)
- 동적 스코프: 함수 호출시에 결정됨
- 정적 스코프 (lexical scope): 함수 정의시 결정됨

```
var x = 'global x';

function outer() {
    var y = 'outer local y';
    console.log(x);
    console.log(y);

    function inner() {
        var x = 'inner local x';
        console.log(x);
        console.log(y);
    }
    inner();
}

outer();
console.log(x);
console.log(y);
```

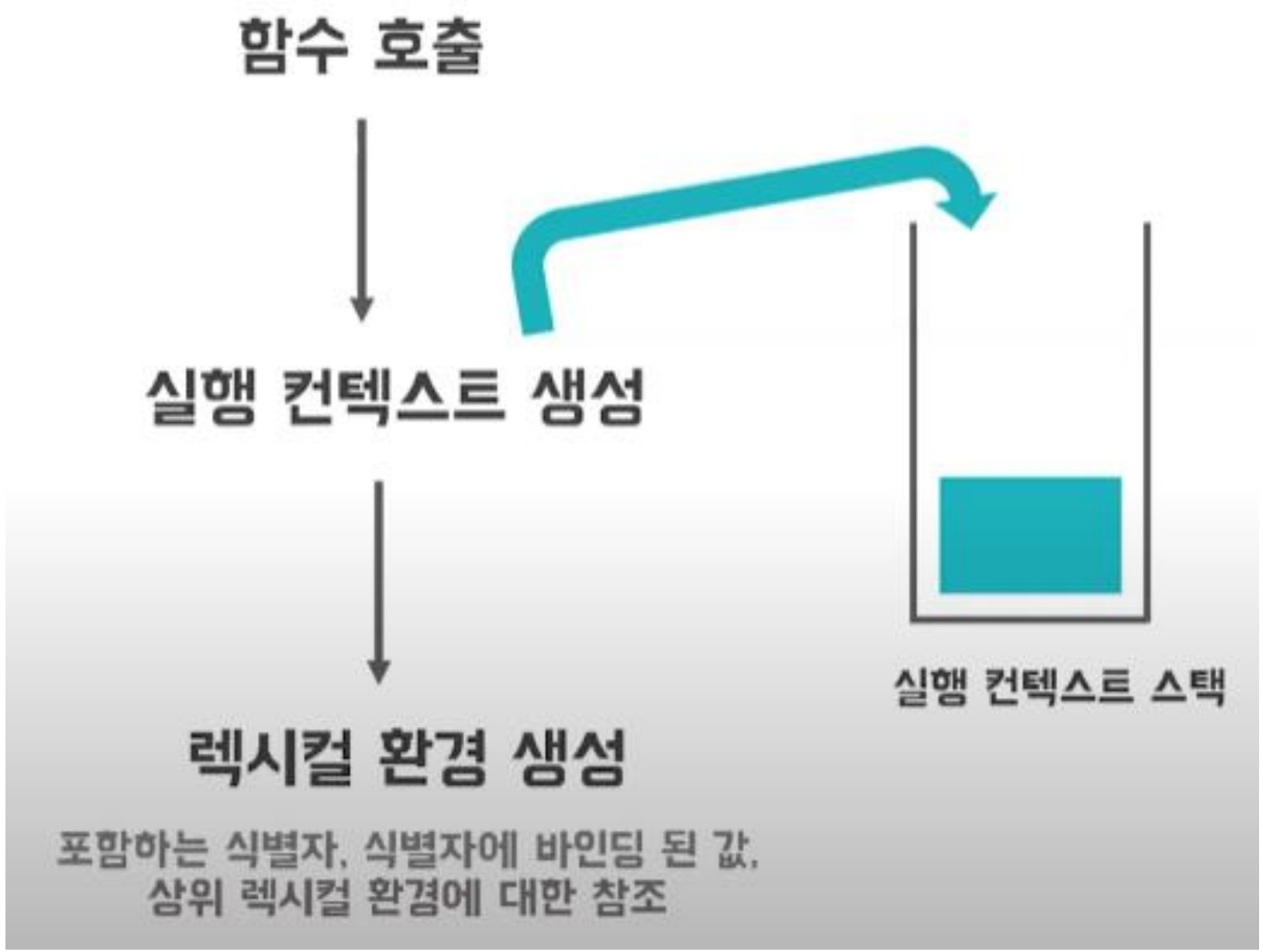


스코프 체인

전역 스코프	
x	나는 전역 x야
outer	<function object>

outer 지역 스코프	
y	나는 outer 함수의 지역 y야
inner	<function object>

inner 지역 스코프	
x	나는 inner 함수의 지역 x야


함수 호출
실행 컨텍스트 생성
실행 컨텍스트 스택
렉시컬 환경 생성
포함하는 식별자, 식별자에 바인딩 된 값,
상위 렉시컬 환경에 대한 참조

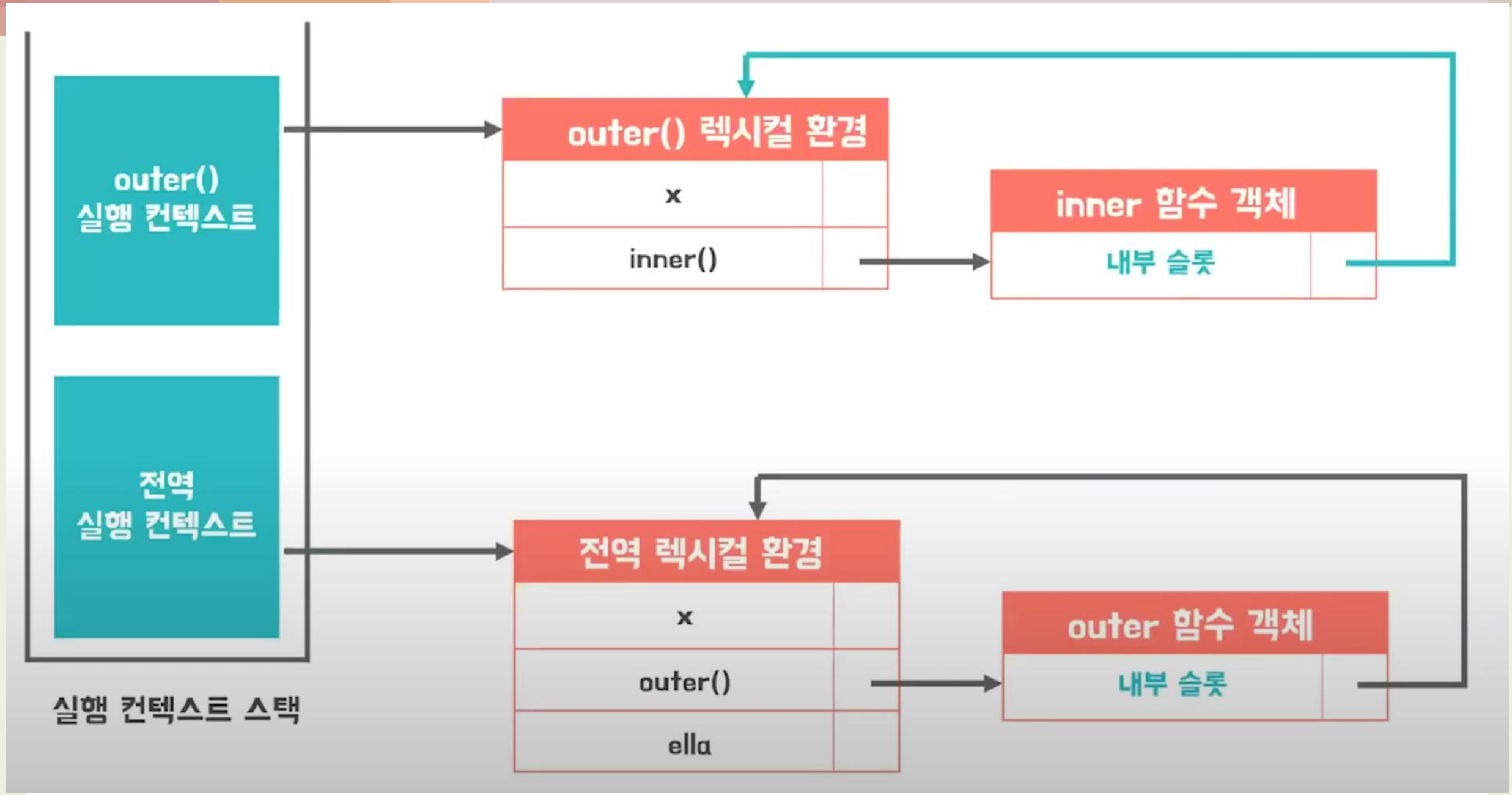
Closure 1

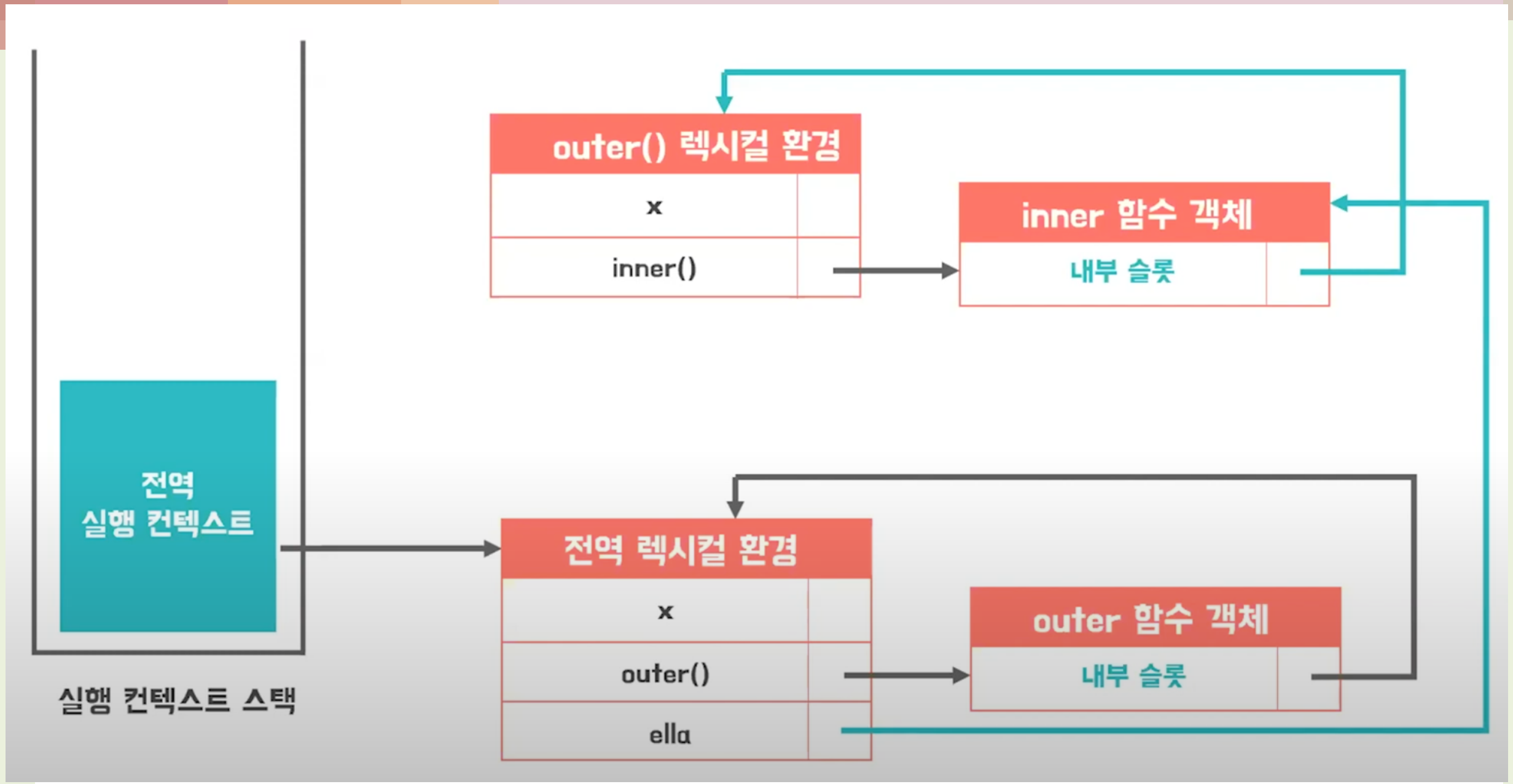
```
const x = 1;

function outer() {
    const x = 10;
    function inner() {
        console.log('x = ', x);
    }

    return inner;
}

var ksd = outer();
ksd();
```


outer()
실행 컨텍스트
outer() 렉시컬 환경
x
inner()
inner 함수 객체
내부 슬롯
전역
실행 컨텍스트
전역 렉시컬 환경
x
outer()
ella
outer 함수 객체
내부 슬롯
실행 컨텍스트 스택

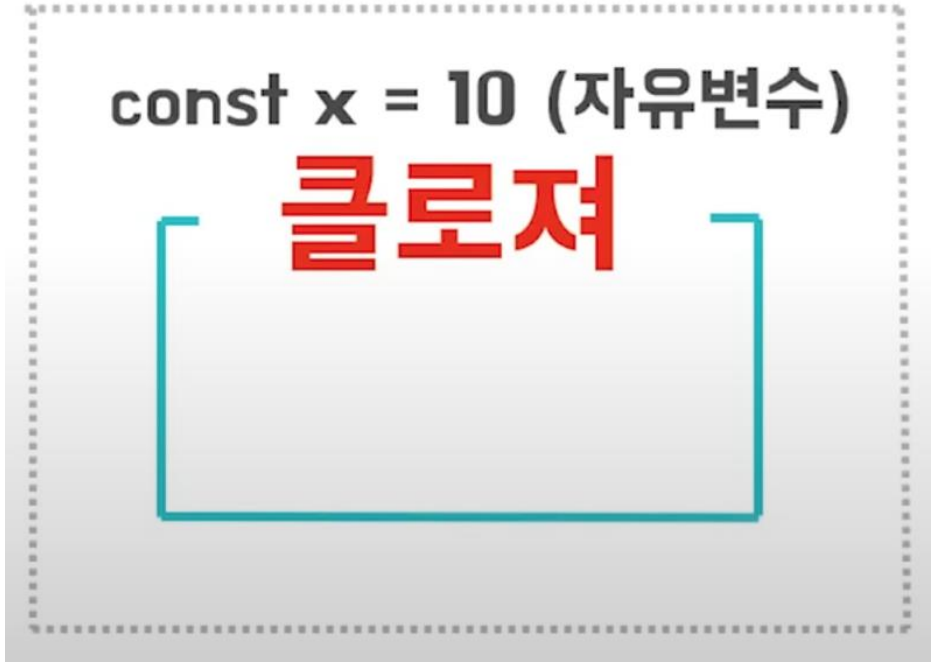

outer() 렉시컬 환경
x
inner()
inner 함수 객체
내부 슬롯
전역
실행 컨텍스트
실행 컨텍스트 스택
전역 렉시컬 환경
x
outer()
ella
outer 함수 객체
내부 슬롯

Closure 1

- 생명주기(life cycle)가 끝난 outer의 내부 변수를 참조할 수 있는 inner() 함수 → closure
- 클로저가 되는 경우: 중첩함수가
  - 상위 스코프의 식별자를 참조하고 있고
  - 외부 함수보다 더 오래 살아 있는 경우
- 자유 변수: 클로저가 참조하는 변수

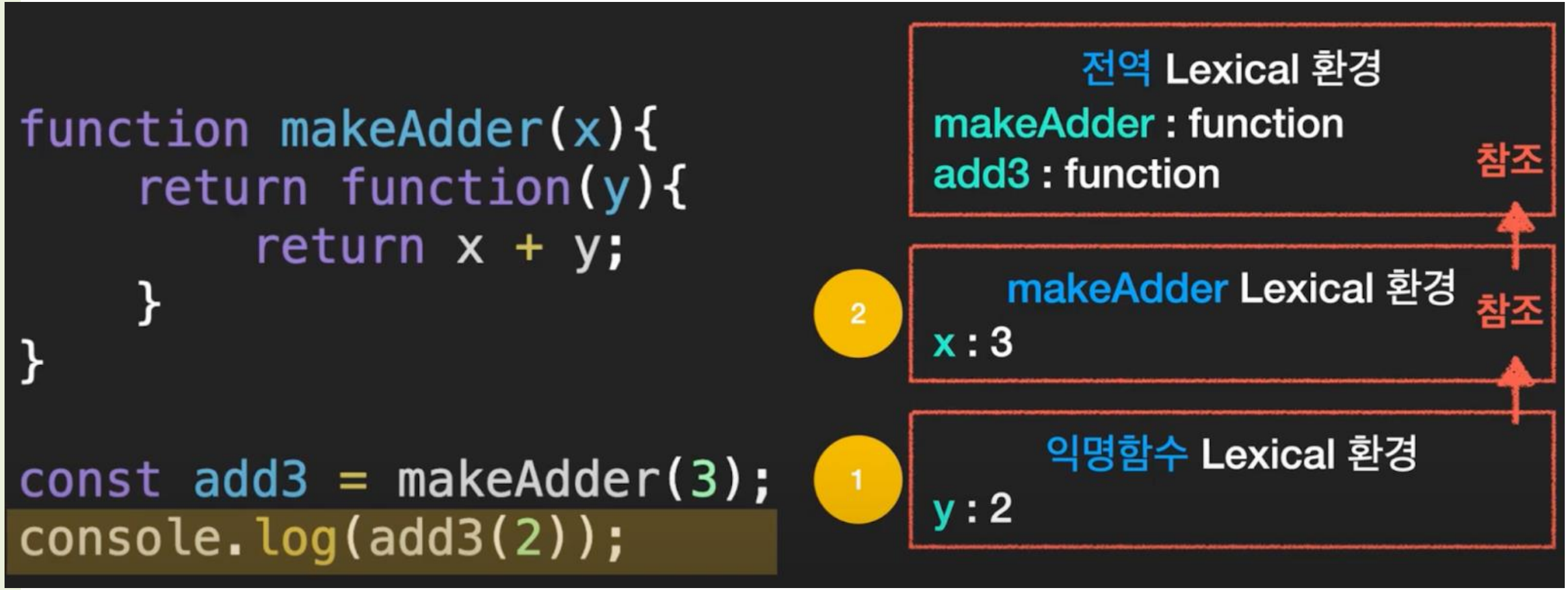
Closure 1

- why closure
  - 하나의 state가 의도치 않게 변경되지 않도록 state를 안전하게 은닉
  - 특정 함수에게만 state 변경을 허용



Closure 2

- What
  - 함수와 렉시컬 환경의 조합
  - 함수가 생성될 당시의 외부 변수를 기억
  - 생성 이후에도 계속 접근 가능
- 함수를 함수 안에 정의하면 내부 함수는 외부 함수의 scope에 접근 가능 → debugger로 확인 !


function makeAdder(x){
return function(y){
return x + y;
}
}
const add3 = makeAdder(3);
console.log(add3(2));
전역 Lexical 환경
makeAdder : function
add3 : function
참조
makeAdder Lexical 환경
x : 3
참조
2
익명함수 Lexical 환경
y : 2
1

```
let l0 = 'l0';
function fn1() {
    function fn2() {
        function fn3() {
            let l3 = 'l3';
            console.log(l0, l1, l2, l3);
        }
        let l2 = 'l2';
        console.log(l0, l1, l2);
        fn3();
    }
        let l1 = 'l1';
        console.log(l0, l1);
        fn2();
    }
    fn1();
```

Closure 2

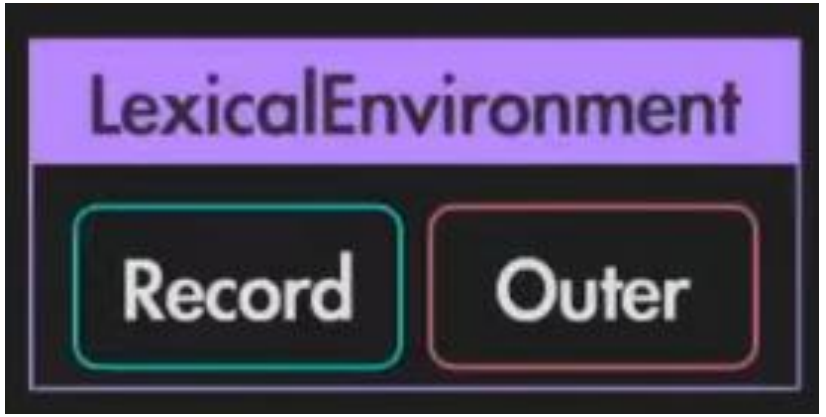
- 활용
  - function 덧셈()에서 '초기값'은 closure에서 접근할 수 있음
  - 함수가 만들어진 시점에, 함수의 부모 함수가 가지는 scope을 함수가 접근가능 => **closure**

```
function 더하기함수공장(초기값) {
    function 덧셈(숫자) {
        return 초기값 + 숫자;
    }
    return 덧셈;
}
```

```
let 더하기1 = 더하기함수공장(1);
console.log(더하기1(1));
console.log(더하기1(2));
let 더하기2 = 더하기함수공장(2);
console.log(더하기2(1));
console.log(더하기2(2));
```

Closure 2

- 실행 컨텍스트
  - LexicalEnvironment
  - LexicalEnvironmentRecord → Record
  - OuterEnvironmentReference → Outer

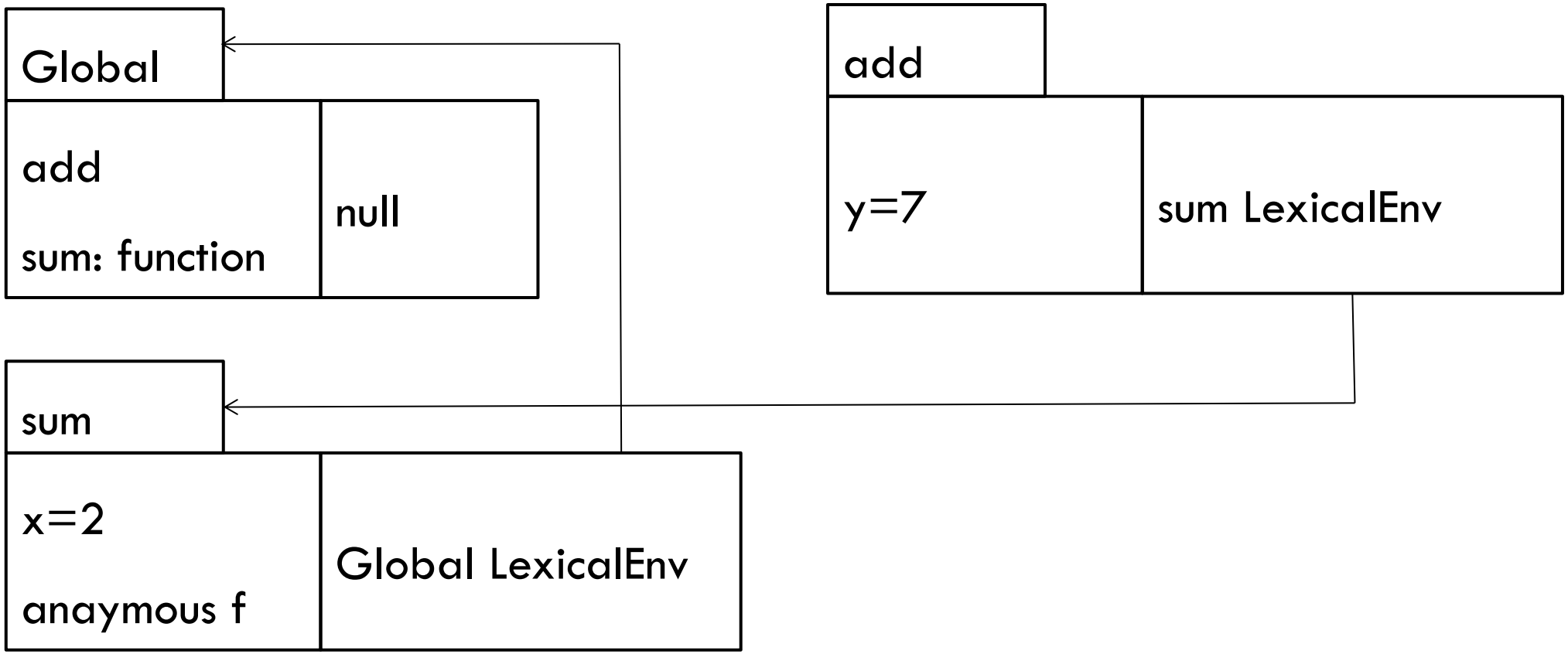


Closure 2

- 정의
  - MDN - 함수와 함수가 선언된 어휘적 환경의 조합
  - 코어 자바스크립트 - 어떤 함수 A에서 선언한 변수 a를 참조하는 내부 함수 B를 외부로 전달할 경우, A의 실행 컨텍스트가 종료된 이후에도 변수 a가 사라지지 않는 현상

```
function doSomething {
    const x = 10;
    function sum(y) {
        return x + y;
    }
    return sum;
}
const something = doSomething();
console.log(something(3));
```

```
function sum(x) {
    return function(y) {
        return x + y;
    };
}
const add = sum(2);
console.log(add(7));
```


Global
add
sum: function
null
add
y=7
sum LexicalEnv
sum
x=2
anaymous f
Global LexicalEnv

Closure 3

- 클로저 – 내부 함수가 외부 함수의 문맥(context)에 접근할 수 있는 것

```
function outer() {
    function inner() {
        let title = 'Web Programming';
        alert(title);
    }
    inner();
}
outer();
```

```
function outer() {
    let title = 'Web Programming';
    function inner() {
    alert(title);
    }
    inner();
}
outer();
```

Closure 3

- 외부 함수가 더 이상 사용되지 않는 경우에도, 내부 함수는 외부 함수에 접근할 수 있음

```
function outer() {
    let title = 'Web Programming';
    return function inner() {
        alert(title);
    }
}
let inner = outer();
inner();
```

Closure 3

- private variable
  - 아무나 변경할 수 없음 → 캡슐화 !
  - title → get(), set()으로만 접금하도록
  - ghost와 matrix는 서로 다른 문맥을 가짐

```
function factory_movie(title) {
    return {
        get_title: function() {
            return title;
        },
        set_title: function(_title) {
            title = _title;
        }
    }
}
```

```
let ghost = factory_movie('Ghost in the shell');
let matrix = factory_movie('Matrix');
```

Source

- Scope & closure 1 - https://www.youtube.com/watch?v=PVYjfrgZhtU
- Closure 2
  - https://www.youtube.com/watch?v=tpI2oXQkGZs
  - https://www.youtube.com/watch?v=bwwaSwf7vkE&t=9s
  - https://www.youtube.com/watch?v=PJjPVfQO61o&t=52s
- Clousure 3 - https://www.youtube.com/watch?v=L8OvfMfIWa0