클로저

클로저

클로저의 정의

MDN

- 정의 - 함수와 함수가 선언된 렉시컬 환경의 조합

코어 자바스크립트

- 정의 - 어떤 함수 A에서 선언한 변수 a를 참조하는 내부 함수 B를 외부로 전달할 경우, A의 실행 컨텍스트가 종료된 이후에도 변수 a 가 사라지지 않는 현상

내가 이해한 클로저

- 정의 - 자신이 선언될 당시의 환경을 기억하는 함수

클로저와 친해지기

출처 : [10분 테크톡] 꼬재의 클로저

클로저 예제 코드

```
function sum(x){
  return function (y) {
    return x+y;
  };
}

const add = sum(1);
console.log(add(2));
```

클로저 동작방식

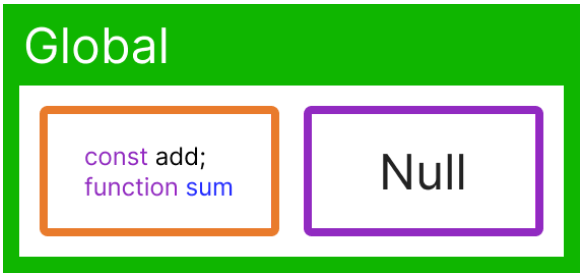
클로저의 동작 방식을 이해하기 위해 사용할 실행 컨텍스트

```
Lexical Environment - Lexical Environment Record
                    ㄴ Outer Environment Reference
```

Lexical Environment
Lexical Environment Record
Outer Lexical Reference

1. 코드 평가 진행 (Global)

   1-1. Global 에 Lexical Environment 생성 후 전역코드 평가



   1-2. Lexical Environment Record 에 add 변수와 sum 함수 호이스팅

   1-3. call 스택에 Global Lexical Environment 푸쉬

2. add 변수에 할당된 sum 함수 실행

   2-1. sum 함수 에 Lexical Environment 생성 후 전역코드 평가

   2-2. Lexical Environment Record 에 sum 함수의 매개변수 x 와 return 문의 익명함수 호이스팅

   2-3. Outer Environment Reference 에는 상위 스코프인 Global Lexical Environment 를 가리킴

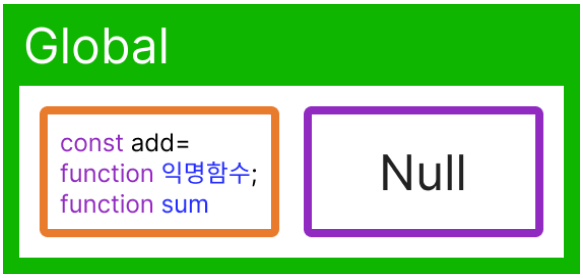
   2-4. Call 스택에 sum Lexical Environment 푸쉬

2-5. 코드 평가 종료후, 매개변수 x 에 할당된 값인 1을 Lexical Environment Record 에 저장

2-6. 더이상 실행할 코드 없기 때문에 call stack 에서 pop

3. 다시 Global Lexical Environment 로 돌아온후, add 함수에 sum함수의 리턴값인 익명함수 할당

   3-1. Global Lexical Environment Record 에 익명함수 할당



4. 마지막 코드인 console.log 를 해결하기 위해, add 함수 호출

   4-1. add Lexical Environment 생성 후 코드평가

   4-2. 매개변수 y 를 호이스팅하여 add Lexical Environment Record 에 저장

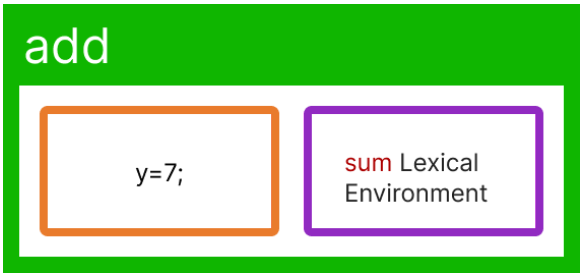
   4-3. Outer Environment Reference 에는 상위 스코프인 sum Lexical Environment 를 가리킴

→ 이때, 클로저 현상이 나타납니다! 분명 sum 은 pop 되어있는 것으로 알고있어 가리킬수 없는것으로 보인다. 그러나 초기에 코드 평가를 할때, 익명함수의 선언과 동시에

Outer Environment Reference 에 먼저 sum 의 Lexical Environmentd 저장하였기 때문에 가리킬수있다.

4-4. Call 스택에 add Lexical Environment 푸쉬 후 코드평가 종료

4-5. 코드 한줄씩 실행 하여, y 매개변수에 2를 할당



4-6. 다음으로 return 문인 x+y 실행

4-7. x의 매개변수가 Lexical Environment Record 에 보이지않기 때문에 Outer Environment Reference 의 Lexical Environment Record 를 찾아 나선다.

4-8. 존재하지 않는 다면 4-7 과정을 반복하여 Global Lexical Environment 까지 탐색한다. (만약 Global Lexical Environment 에도 존재하지 않는 다면 reference 에러가 발생한다.)

4-9. sum Lexical Environment 에 할당한 x 값이 존재하기 때문에 그 값을 확인후 본래의 값으로 돌아갑니다.

4-10. 계산된 값 (1 + 2) 인 3의 값이 리턴됩니다.

4-11. 더이상 실행할 코드 없기 때문에 call stack 에서 add Lexical Environment 를 pop

4-12. Global Lexical Environment 에서 마지막으로 남은 console.log(3)을 출력한다.

4-13. Global Lexical Environment 도 call stack 에서 pop 되면서 프로그램 종료

클로저 장/단점

장점

- 변수와 함수를 캡슐화하여 전역변수의 방해를 방지
- 코드의 가독성과 유지보수성을 높일 수 있음
- 외부 변수에 접근하여 값을 변경하거나 읽어올 수 있기 때문에, 비동기 처리나 콜백 함수에서 상태를 유지하기 좋음

단점

- 변수를 변경하거나 함수를 호출하는 등의 작업이 많아지면, 코드의 복잡도가 증가
- 남발하게 된다면 메모리 누수가 발생