24장. 클로저

24.1 렉시컬 스코프

▼ 렉시컬 스코프란?

함수를 어디서 호출했는지가 아니라 함수를 어디에 정의했는지에 따라 상위 스코프를 결정

→ 렉시컬 스코프(= 정적 스코프)

→함수의 상위 스코프는 함수를 정의한 위치에 의해 정적으로 결정되고 변하지 않음

▼ 스코프 체인이란?

자신의 “외부 렉시컬 환경에 대한 참조”를 통해 상위 렉시컬 환경과 연결되는 것

24.2 함수 객체의 내부 슬롯

▼ 렉시컬 스코프가 가능하려면?

자신의 내부 슬롯 [[Environment]]에 자신이 정의된 환경, 즉 상위 스코프의 참조를 저장

→ 이때 상위 스코프의 참조는 현재 실행 중인 실행 컨텍스트의 렉시컬 환경을 가리킨다.

24.3 클로저와 렉시컬 환경

▼ 클로저란?

외부 함수보다 중첩 함수가 더 오래 유지되는 경우 중첩 함수는 이미 생명 주기가 종료한 외부 함수의 변수를 참조할 수 있는데 이러한 중첩 함수를 클로저라고 부른다.

→ 자바스크립트의 모든 함수는 상위 스코프를 기억하므로 이론적으로 모든 함수는 클로저이다.

▼ 클로저가 아닌 경우

1. 어떤 식별자도 참조하지 않는 함수는 클로저가 아니다.

2. 외부 함수보다 일찍 소멸되면 클로저가 아니다.

▼ 자유 변수란?

클로저에 의해 참조되는 상위 스코프의 변수

24.4 클로저의 활용

▼ 클로저 사용이유

상태를 안전하게 변경하고 유지하기 위해 사용

→ 상태를 안전하게 은닉하고 특정 함수에게만 상태 변경을 허용하기 위해 사용

24.5 캡슐화와 정보 은닉

▼ 캡슐화란?

객체의 상태를 나타내는 프로퍼티와 프로퍼티를 참조하고 조작할 수 있는 동작인 메서드를 하나로 묶는 것을 말한다.

▼ 정보은닉이란?

캡슐화를 객체의 특정 프로퍼티나 메서드를 감출 목적으로 사용하는 것

24.6 자주 발생하는 실수

▼ 키워드 사용

var대신 let, const 키워드 사용하기