closure

- 원리
  - 내부함수는 외부함수의 지역변수에 접근 할 수 있는데 외부함수의 실행이 끝나서 외부함수가 소멸된 이후에도 내부함수가 외부함수의 변수에 접근 할 수 있다.

    이러한 메커니즘을 클로저라고 한다.
- 사용이유
  1. 현재 상태를 기억하고 변경된 최신 상태를 유지하기 위해
  2. 전역 변수의 사용을 억제 하기 위해
  3. 정보를 은닉하기 위해
- 예시

```
function outer() {
  var name = `closure`;
  function inner() {
    console.log(name);
  }
  inner();
}
outer();
// console> closure
```

  - `outer` 함수를 실행시키는 `context` 에는 `name` 이라는 변수가 존재하지 않는다는 것을 확인할 수 있다. 비슷한 맥락에서 코드를 조금 변경해볼 수 있다.

```
var name = `Warning`;
function outer() {
  var name = `closure`;
  return function inner() {
    console.log(name);
  };
}

var callFunc = outer();
callFunc();
// console> closure
```

위 코드에서 `callFunc` 를 클로저라고 한다. `callFunc` 호출에 의해 `name` 이라는 값이 console 에 찍히는데, 찍히는 값은 `Warning` 이 아니라 `closure` 라는 값이다.

즉, `outer` 함수의 context 에 속해있는 변수를 참조하는 것이다. 여기서 `outer`함수의 지역변수로 존재하는 `name`변수를 `free variable(자유변수)`라고 한다.

- 이처럼 외부 함수 호출이 종료되더라도 외부 함수의 지역 변수 및 변수 스코프 객체의 체인 관계를 유지할 수 있는 구조를 클로저라고 한다. 보다 정확히는 외부 함수에 의해 반환되는 내부 함수를 가리키는 말이다.