

2023.05.07 D.P.



지독하게  
빠져든다...

아직도 잘 모르겠는

클로저

CLOSURE



2023-05-07

# 클로저 (CLOSURE)

: 무엇이고, 어디에 쓰이나요

D.P. 김도경

**클로저는 무엇이고, 어디에 쓰이나요?**

# 클로저는 무엇이고, 어디에 쓰이나요?

**클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.**

A closure is the combination of a function and the lexical environment within which that function was declared.

**클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.**

클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.

```
> const x = 1;
```

```
function outerFunc() {  
    const x = 10;  
    innerFunc();  
}
```

```
function innerFunc() {  
    console.log(x);  
}
```

```
outerFunc();  1 출력
```

클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.

**자바스크립트 엔진은**


**함수를 어디서 호출**했는지가 아니라

**함수를 어디에 정의**했는지에 따라

**상위 스코프를 결정한다.**

**이를 렉시컬 스코프라 한다.**

클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.

```
const x = 1;  
  
function outerFunc() {  
    const x = 10;  
    const innerFunc() {  
        console.log(x);  
    }  
  
    return innerFunc;  
}  
  
const inner = outerFunc();  
inner();  10 출력
```



외부 함수보다 중첩 함수가 **더 오래 유지되는 경우**

**중첩 함수는**

**이미 생명 주기가 종료한 외부 함수의 변수를 참조할 수 있다.**

**이러한 중첩함수를 클로저라고 부른다.**

turn

된다.





## 클로저:

어떤 함수 A에서 선언한 변수  $\alpha$ 를 참조하는 내부함수 B를  
외부로 전달할 경우

**A의 실행 컨텍스트가 종료된 이후에도 변수  $\alpha$ 가 사라지지 않는 현상**

turn

된다.



## 클로저:

어떤 함수 A에서 선언한 변수  $a$ 를 참조하는 내부함수 B를  
외부로 전달할 경우

**A의 실행 컨텍스트가 종료된 이후에도 변수  $a$ 가 사라지지 않는 현상**

turn

된다.

# 클로저는 무엇이고, 어디에 쓰이나요?

**클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.**

A closure is the combination of a function and the lexical environment within which that function was declared.

**클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.**

클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.

```
function sum(a, b) {  
  console.log(a + b);  
}  
  
function getClosure(func, a, b) {  
  return function fn() {  
    func(a, b);  
  };  
}  
  
const myFunc = getClosure(sum, 10, 20);  
setTimeout(myFunc, 1000);
```

setTimeout과 같은 콜백 함수를 받는 함수에서 인자를 넘기고 싶은 경우

클로저는 콜백함수, 모듈패턴, 커링 함수 등에서 사용할 수 있다.



대부분의 객체지향 프로그래밍 언어는 클래스를 정의하고 public private protected 와 같은 접근 제한자를 선언하여 공개 범위를 한정할 수 있다.

하지만 자바스크립트는 접근 제한자를 제공하지 않는다.  
그래서 기본적으로 외부에서의 참조를 막을 수 없다.

이때 클로저를 활용해서 private 변수를 구현할 수 있다.



클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.

```
var myCounterModule = (function() {  
  var counter = 0;  
  return {  
    increase: function() {  
      return (++counter);  
    },  
    decrease: function() {  
      return (--counter);  
    },  
    getCounter: function() {  
      return counter;  
    }  
  }  
})();
```

```
> myCounterModule.increase();  
< 1  
-----  
> myCounterModule.increase();  
< 2  
-----  
> myCounterModule.decrease();  
< 1
```

외부에서는 counter라는 변수에 직접 접근을 할 수 없다.

오직 **메소드를 통해서만** 값을 더하거나 뺄 수 있다.

클루저는 콜백함수, 모듈패턴, 커링 함수 등에서 사용할 수 있다.



## 커링 함수:

여러개의 인자를 받는 함수를 하나의 인자만 받는 함수로 나눠서  
순차적으로 호출될 수 있게 체인 형태로 구성한 것.

클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.

```
function multiply(x, y, z){  
    return x, y, z;  
}  
  
console.log(multiply(3, 5, 7));  
console.log(multiply(3, 5, 8));  
console.log(multiply(3, 2, 1));
```

함수에 모든 인자를 직접 지정해주어야 한다.

클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.

```
function multiply(x) {  
  return function(y) {  
    return function(z){  
      return x*y*z;  
    }  
  }  
}
```

```
let multiply3 = multiply(3);    // 3이 고정됨  
let multiply3And5 = multiply3(5); // 3과 5가 고정됨  
console.log(multiply3And5(7));  
console.log(multiply3And5(8));  
console.log(multiply3(2)(1));
```

x만 먼저 받고, 나중에 y, z를 받을 수 있다.

(특정 인자를 재사용할 수 있다.)

클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.

```
function multiply(x, y, z){  
  return x, y, z;  
}  
  
console.log(multiply(3, 5, 7));  
console.log(multiply(3, 5, 8));  
console.log(multiply(3, 2, 1));
```

```
function multiply(x) {  
  return function(y) {  
    return function(z){  
      return x*y*z;  
    }  
  }  
}  
  
let multiply3 = multiply(3);    // 3이 고정됨  
let multiply3And5 = multiply3(5); // 3과 5가 고정됨  
let multiply3And2 = multiply3(2); // 3과 2가 고정됨  
console.log(multiply3And5(7));  
console.log(multiply3And5(8));  
console.log(multiply3And2(1));
```





## 참고

클로저의 단점으로 일컬어지는 메모리 누수의 위험을 조심하기 위해서  
사용이 끝난 시점에는 의도적으로 참조 카운트를 0으로 만들면서  
메모리가 회수될 수 있게 해야합니다.

(Ex, 식별자에 null 이나 undefined 할당)

# 클로저는 무엇이고, 어디에 쓰이나요?

**클로저는 함수와 그 함수가 선언된 렉시컬 환경과의 조합이다.**

A closure is the combination of a function and the lexical environment within which that function was declared.

**클로저는 콜백함수, 모듈패턴, 커링 함수, 등에서 사용할 수 있다.**

2023-05-07

**감사합니다.**

**D.P. 김도경**