# 9강\_함수-심화

- 9-2 클로저
- 9-3 내부함수 9-4 콜백함수
- 9-5 내장함수

#### 9-1: 함수 반환(리턴)함수

Ex: 9\_01.html

# 함수는 또 다른 함수를 반환할 수 있다.

```
function funName() {
실행문

return function(){
실행문
};

}
```

#### 9-2: 클로저

Ex: 9\_02.html

# 반환된 함수의 life-cycle가 종료될 때까지 호출된 함수의 life-cycle이 유지되는 것

## scop

```
function funName(x) {

var varName = ":: 요청한 " + x + "에 대한 구구단::";

return function() {

console.log(varName);

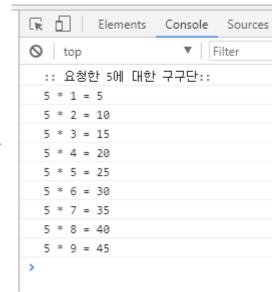
for(var i=1; i<10; i++) {

console.log(x + " * " + i + " = " + (x*i));

};

var returnFun = funName(5);

returnFun();
```



### 9-3: 내부함수

Ex: 9\_03.html

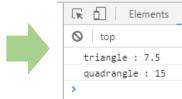
# 함수안에 또다른 함수

```
function outFun () {

innerFun1() {...}
innerFun2() {...}
...
innerFunn() {...}
}

내부 함수로 외부에서는 호출 불가
```

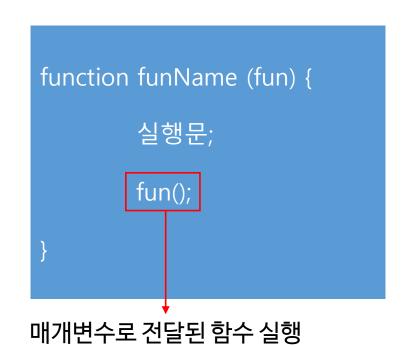
```
내부함수
    : 함수안에 또다른 함수
function outFun(width, height) {
   console.log("triangle : " + triangle());
   console.log("quadrangle : " + quadrangle());
   //삼각형 면적 계산
   function triangle() {
       return (width * height) / 2;
   //사각형 면적 계산
   function quadrangle() {
       return width * height;
outFun(3, 5);
```



#### 9-4: 콜백함수

Ex: 9\_04.html

## 매개변수로 함수를 전달하고, 전달된 매개변수가 특정 시점에 호출되는 것



```
콜백함수
    : 매개변수로 함수를 전달
function funName (funCBF, funCBP, num) {
    for(var i=1; i<=10; i++) {
        console.log(num + " * " + i + " = " + (num * i));
        if(i < 10) funCBP(i);
    funCBF();
function funCallBackProgress(n) {
    console.log((n * 10) + "% 진행 완료!");
};
function funCallBackFinish() {
    console.log("서버 작업 종료!");
};
funName(funCallBackFinish, funCallBackProgress, 7);
```

```
Elements
♦ top
 7 * 1 = 7
 10% 진행 완료!
 7 * 2 = 14
 20% 진행 완료!
 7 * 3 = 21
 30% 진행 완료!
 7 * 4 = 28
 40% 진행 완료!
 7 * 5 = 35
 50% 진행 완료!
 7 * 6 = 42
 60% 진행 완료!
 7 * 7 = 49
 70% 진행 완료!
 7 * 8 = 56
 80% 진행 완료!
 7 * 9 = 63
 90% 진행 완료!
 7 * 10 = 70
 서버 작업 종료!
```

9-5: 내장함수

Ex: 9\_05.html

개발자가 직접 만들지 않아도 javascript가 기본적으로 제공해주는 함수

alert();

setTimeout();

clearTimeout();

console.log();

|setInterval();

eval();

clearInterval();