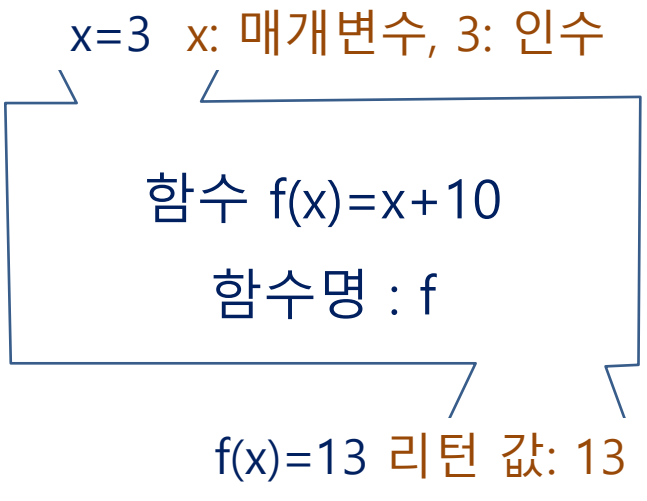
04 함수

- 함수 정의
- 함수 선언과 호출

❖ 함수(function) : 특정 기능을 수행하는 코드들의 집합




```
function f(x){
    return x+10;
}
console.log(f(3));
```

함수도 하나의 자료형

함수 선언

```
function 함수명(매개변수){
    함수코드
    return();
}
```

함수실행(호출 한다)

```
함수명(인수:매개변수 값);
```

❖ 함수(function) 선언

함수 정의 방식1 : 선언문 형태 – 이름이 있는 선언적 함수

```
function myFunc(param){
  console.log(param + " run!");
}
```

함수 정의 방식2 : 함수 리터럴 – 이름이 없는 익명함수

```
const myFunc2 = function(param){
  console.log(param + " run!");
}
```

함수 사용 : 호출

```
myFunc("func1");
myFunc2("func2");
```

❖ 매개변수와 리턴값

- 매개변수의 개수가 맞지 않아도 허용
  - 추가된 매개변수 : 무시
  - 지정되지 않은 매개변수 : undefined 입력

```
function sum(a=0, b=0){    => 기본값 지정
  return a+b;
}
console.log(sum(4,5));
console.log(sum(4));
console.log(sum());
```

- return : 함수 실행도중 호출한 곳으로 돌아가라는 의미
  - 아무 값도 리턴하지 않은 경우 자료형과 값 모두 undefined

❖ 함수(function) 종류

- 함수 사용의 목적은 코드 재사용성
- 익명함수 : 이름이 없는 함수

  function (){ };

  let 함수명 = function (){ } : 변수에 넣어 사용
- 선언적 함수 : 이름이 있는 함수

  function 함수명( ){ }
- 자기 호출 함수(즉시실행) : 함수를 정의함과 동시에 실행하는 함수

  (function() { }) ( ); → 변수에 할당 가능
- 콜백 함수 : 다른 함수의 매개변수로 전달하는 함수
- 화살표 함수 : 함수 선언을 간단하게 작성

  ( ) => { }

❖ 익명 함수와 선언적 함수

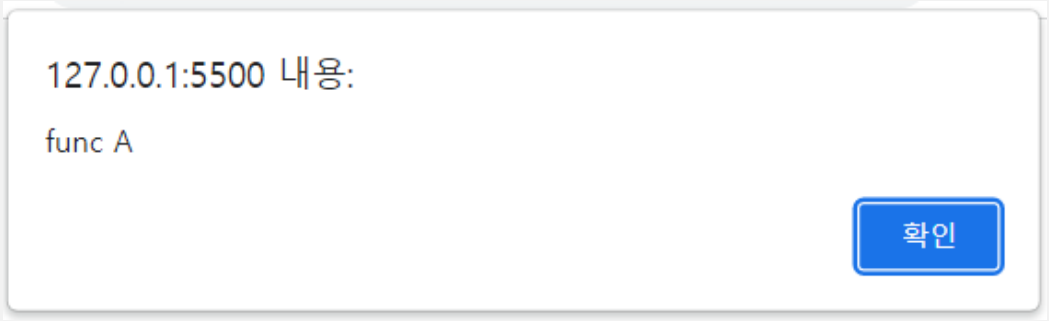
```
// 익명함수 – 선언 전에 호출하면 오류 발생
    func();
    var func = function(){alert('func A')};
    var func = function(){alert('func B')};

// 선언적 함수 - 함수 선언이 먼저 실행 되므로 호출 가능
    func(); - ③
    function func(){alert('func A')};          - ①
    function func(){alert('func B')};          - ②
// 실행순서  ①②③, 선언함수
```

❖ 익명 함수와 선언적 함수

```
// 함수 생성
    var func = function(){alert('func A')};  - ②
    function func(){alert('func B')};        - ①
// 함수호출
    func(); - ③

// 실행 결과는?
```



❖ 화살표 함수(익명함수에서만 사용)

```
let sum =  function(a, b){
   return a + b;
 }

console.log(sum(5, 10));

const hi = function(){
    return “hi?”;
 }

let hi = function(user){
   alert(‘hi?’);
}
```

```
let sum = (a, b) => {return a + b}

let sum = (a , b) => a + b;

console.log(sum(5, 10));

const hi = () => {return “hi?”}
const hi = () => “hi?”


let hi = (user) => alert(‘hi?’);
```

❖ 내부함수

- 다른 개발자와 충돌을 방지 하기 위해 사용(함수명, 변수 등)
- pyta() 함수 외부에서 내부함수 square() 함수를 사용할 수 없다

```
function pyta(width, height){
        function square(x){
          return x*x;
        }
        return Math.sqrt(square(width)+square(height));
    }
    let result = pyta(4, 9);
    console.log(result);
```

❖ 콜백 함수

- 함수를 매개변수로 전달되는 함수
- 함수도 하나의 자료형이므로 전달가능

```
function callTenTimes(callback){
        for ( let i = 0; i<10; i++){
          callback();
        }
    }
        let callback = function(){
        console.log('함수호출')
     }
     callTenTimes(callback);
```

❖ 리턴되는 함수

- 함수의 결과 값으로 리턴 되는 함수

```
function returnFunction(){
    return function(){
      console.log('hello Function');
    };
  }
  returnFunction()();
```

```
returnfunction()();
⇒ ( function(){ console.log('hello Function')} )();
```

❖ 리턴 함수

```
function test(name){
 let output = 'hello! ' + name;

 return function(){
        return output;
 };
}
let test1 = test('web');
let test2 = test('javascript');
console.log(test1());
console.log(test2());
```

```
test1() : (function(){
        return output;
    })();

test2() : (function(){
        return output;
    })();
```

❖ 클로저 : 만들어진 시점의 실행환경을 기억하는 함수

- 지역변수는 함수 외부에서 사용할 수 없고, 함수가 종료되면 제거
- 클로저 : - 지역변수를 남겨두는 현상, 남겨진 지역변수(자유변수)
  - 내부 변수들이 남아있는 공간,
  - 리턴된 함수 자체

- 반드시 리턴된 클로저 함수를 통해 지역변수 사용(지역변수 보호)
- 클로저는 변수의 값이 의도치 않게 변경되지 않도록 은닉하고,
  특정 함수에게만 변경을 허용하여,
  안전하게 변경 유지하기 위해 사용

❖ 클로저 : 만들어진 시점의 실행환경을 기억하는 함수

```
let increse =  function(){
    let num = 0;
    return ++num;
}


console.log(increse());
console.log(increse());
 console.log(increse());
```

```
let increse =  (function(){
        let num = 0;
        return function(){
            return ++num;
        }
    }());
    console.log(increse());
    console.log(increse());
    console.log(increse());
```

❖ 펼침연산자(spred operator)

```
let calc =function(x, y, ...restparams){   → 나머지 파라메터는 맨 마지막에 위치
  return x + y + restparams.reduce(function(sum, param){
                                    return sum + param;
                                    });
}
let arr = [0,1];
console.log(calc(-1, ...arr,  2, ...[3])); → 전개구문 : 펼침 연산자로 작성한 코드
console.log(calc(-1,0,1,2,3));
let arr2=[1,2,3,4,5,6];
console.log(calc(...arr2));
console.log(calc(null, ...arr2));
```

... 펼침 연산자 : 변수명 앞에 ... 표시
→ 요소를 펼쳐서 각각의 개별요소로 적용

❖ 변수의 스코프(scope)

```
var a=1;
var b=5;

function outfunc(){
        function innerfunc(){
            a=b;
        }
  console.log(a);

  a=3;
  b=4;
  innerfunc();
  console.log(a);
  var b= 2; console.log(b);
}

outfunc(); console.log(b);
```

전역 a=1, b=5

출력 a=1

전역 a=3, 지역 b=4
전역 a=4, 지역 b=4

출력 a=4
지역 b=2

----- ?

---------?

❖ 함수 용어 정리

용어	설명
함수	함수는 하나의 자료형
호출	선언된 함수의 내부 코드를 실행하는 것
매개변수	함수 내부로 자료 값을 넘기기 위해 사용하는 변수
리턴	함수 실행 결과값을 호출한 곳으로 넘기는 것
인수	매개변수로 전달되는 값
익명함수	이름 없이 선언된 함수, 변수에 대입가능
선언적 함수	이름이 있는 함수
콜백 함수	매개변수로 전달되는 함수
클로저	만들어진 시점의 실행환경을 기억하는 함수

❖ 도전! 문제

문제1

최소값(min)과 최대값(max)을 넘겨 받아 min 부터 max 까지의 합을 구하고, 계산 결과 합을 리턴 하는 함수를 생성하세요.

함수를 호출해서 결과를 출력하세요.

❖ 도전! 문제

문제2 이름, 국어, 영어, 수학 데이터를 넘겨 받아 성적을 처리하는 함수를 생성 하세요

함수호출

```
sungjuk('aaa', 80, 75, 90);
sungjuk('bbb', 90, 88, 99);
sungjuk('ccc', 100, 85, 94);
```

출력모양

이름	총점	평균	등급
aaa	245	82	B
bbb	277	92	A
ccc	279	93	A

avg = Math.round(sum/3) : 평균은 반올림해서 정수로 계산