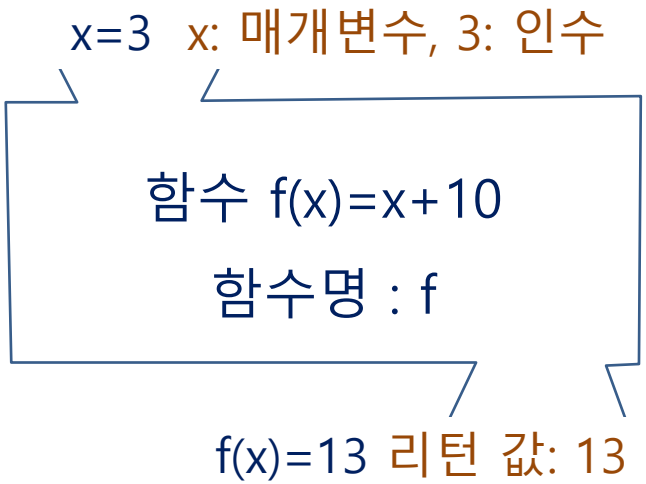
04 함수와 이벤트

- 함수 정의
- 함수 선언과 호출
- 이벤트 처리

❖ 함수(function) : 특정 기능을 수행하는 코드들의 집합

- 함수 정의 : 프로그램을 기능별로 묶어 놓은 덩어리
- 명령의 시작과 끝을 명확하게 구별 가능
- 같은 기능이 필요한 곳에 재사용
- 내장 함수 : 자바스크립트에서 미리 만들어 놓은 함수
- 사용자 정의 함수 : 개발자가 필요한 기능을 직접 만드는 함수




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

❖ 함수(function) 선언

- 함수 선언(정의) : 함수가 어떤 명령을 처리할 지 알려 주는 것
- 함수 호출(실행) : 선언한 함수를 사용하는 것

함수 선언

```
function 함수명(매개변수){
        명령 코드
        return();
}
```

* 함수 선언 부분이 가장 먼저 해석
 -> 선언된 위치에 상관없이 실행 가능
* 여러 개의 함수를 앞이나 뒷부분에 모아 두고 필요할 때 호출해서 사용

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!");
}
```

변수명

- 변수에 저장해서 사용
- 하나의 식으로 사용

❖ 함수 사용 : 호출

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

❖ 변수의 스코프

- **스코프(유효 범위)** : 변수를 사용할 수 있는 범위
  - 어디에서 선언 되었느냐에 따라 범위 결정
- 지역(로컬) 변수 : 한 함수 안에서만 사용 가능한 변수
- 전역(글로벌) 변수 : 스크립트 전체에서 사용할 수 있는 변수
  - 함수 밖에서 선언 하거나,
  - 함수 안에서 var 예약어를 빼고 선언
  - var로 선언한 변수는 호이스팅, 재선언, 재할당 됨

  => **사용 중인 변수를 재선언, 재할당 하는 실수로 오류 발생**
- **let 선언 변수 : 블록 변수 – 사용 권장**
  - 재선언 불가
  - 호이스팅 없음 – 초기화 전에 사용 불가
- const 선언 : 자주 사용하는 상수 변수에 담아 사용
  - 재선언, 재할당 불가

❖ 매개변수와 리턴값

- 매개변수 : 외부에서 값을 받아 줄 변수
  - 선언된 함수 안에서만 사용
  - 추가된 매개변수 : 무시
  - 지정되지 않은 매개변수 : undefined 입력

```
function sum(a=0, b=0){    => 기본값 지정
  return a+b;
}                          인수(argument) : 함수로 전달할 실제 값
console.log(sum(4,5));
console.log(sum(4));
console.log(sum());
```

- return : 함수를 실행한 위치로 값을 반환
  - 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() { }) ( ); → 한 번만 실행 하는 함수
- 콜백 함수 : 다른 함수의 매개변수로 전달하는 함수
- 화살표 함수 : 함수 선언을 간단하게 작성

  ( ) => { }

❖ 익명 함수와 선언적 함수

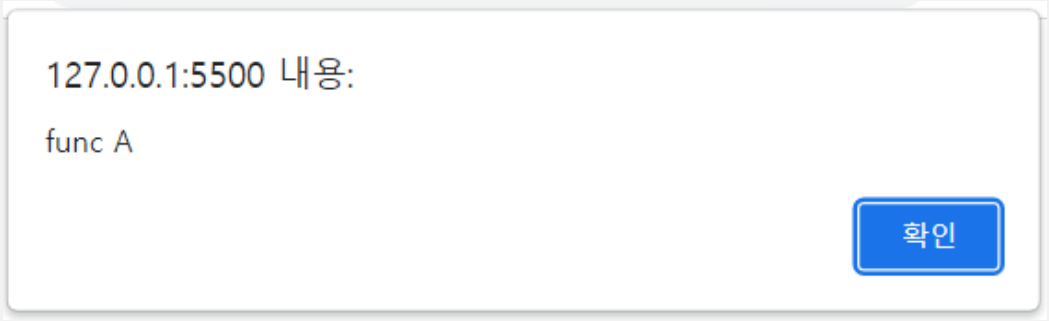
```
// 익명함수 – 선언 전에 호출하면 오류 발생
    func();
    var func = function(){alert('func A')};
    var func = function(){alert('func B')};

// 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

- 함수의 매개변수로 전달되는 함수
- 함수도 하나의 자료형이므로 전달가능

```
function order(coffee, callback){
  console.log(`${coffee} 주문접수`);
  setTimeout(()=>{callback(coffee)},3000);
}
function done(coffee){
      console.log(`${coffee} 준비 완료`);
}
order('아메리카노', done);
```

❖ 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

- 함수 안에서 선언된 변수는 외부에서 사용할 수 없고, 함수가 종료되면 제거됨

- **클로저**는 실행이 종료된 함수의
  환경정보 값(변수 등)을 익명함수(클로저)를 통해 전달해 주는 기능

- **클로저**는 사용자가 임의로 변경하면 안 되는 변수를 안전하게 변경하고 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 increse2 =  (function(){
        let num = 0;
        return function(){
            return ++num;
        }
    }());
    console.log(increse2());
    console.log(increse2());
    console.log(increse2());
```

❖ 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	프로그램에서 기능별로 묶어놓은 명령 덩어리
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 실제 값
익명함수	이름 없이 선언된 함수, 변수에 대입가능
선언적 함수	이름이 있는 함수
콜백 함수	매개변수로 전달되는 함수
클로저	만들어진 시점의 실행환경을 기억하는 함수

❖ 이벤트(event)

- 이벤트 : 웹 브라우저나 사용자가 행하는 어떤 동작
- 웹 문서 안에서 이루어지는 동작에서만 발생
- 키보드, 마우스, 웹 문서 불러올 때, 폼에 내용을 입력할 때 발생

```
Window.onload = function () {
    let header = document.getElementById('header');
    function whenClick(){ alert('CLICK'); }

    header.onClick = whenClick;
};
```

이벤트 이름 : onClick
이벤트 속성

whenClick;
이벤트 처리기
이벤트 핸들러

- **이벤트 모델** : 문서 객체에 이벤트를 연결하는 방법

❖ 이벤트(Event)

마우스 이벤트 발생

속성	설명
click	요소를 마우스로 클릭했을 때
dbclick	요소를 두 번 눌렀을 때
mousemove	요소 위에서 마우스 포인터를 움직일 때
mousedown	요소 위에 마우스를 누르는 동안
mouseup	요소를 누르고 있다가 버튼에서 손을 땔 때
mouseover	요소 위에 마우스가 올려질 때
mouseout	마우스 포인터가 요소를 벗어날 때

❖ 이벤트(Event)

키보드 이벤트 발생

속성	설명
keypress	키를 눌렀을 때
keydown	키를 누르고 있는 동안
keyup	키에서 손을 땔 때

❖ 이벤트(Event)

문서 로딩 이벤트 발생

속성	설명
abort	웹 문서 로딩 중에 멈췄을 때
error	문서가 정확하게 로딩되지 않았을 때
load	문서 로딩이 끝났을 때
resize	문서의 화면 크기가 바뀠을 때
scroll	문서가 스크롤 되었을 때
unload	문서를 벗어날 때

❖ 이벤트(Event)

폼 이벤트 발생

속성	설명
blur	폼 요소에 포커스를 잃었을 때
change	목록이나 체크상태 등이 변경되었을 때
focus	폼 요소에 포커스가 놓였을 때
reset	폼이 다시 시작되었을 때
submit	submit 버튼을 눌렀을 때

❖ 이벤트(Event)

이벤트 처리

- 이벤트가 발생하면 바로 이벤트 처리 함수 연결
- 이벤트 발생은 이벤트 이름 앞에 '**on**' 을 붙여 사용
- 이벤트 처리기 = 실행명령이나 함수 연결
- 태그에 직접 입력하는 **인라인 이벤트 모델**

```
<h1 id="heading" onclick="alert('click ')">이벤트 처리</h1>
```

- **DOM을 이용한 이벤트 연결**

```
document.getElementById('heading').onclick=function(){
   alert('click');
 }
 document.getElementById('heading').onclick= null --- 제거
```

❖ 도전 문제

문제1

- 상세페이지 보기/닫기

상세설명 보기

커피

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Excepturi quaerat minima facere odit pariatur officiis aspernatur quam, facilis assumenda, consequuntur enim est dignissimos sequi voluptatibus aliquam iste aperiam vitae. Nam?

상세설명 닫기

❖ 도전! 문제

문제2

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

문제3 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