05. Function

1. 함수 선언 및 호출

```
// 함수 선언
function doSomething() {
   console.log('hello');
}
```

2. 함수를 인자로 전달

```
function doSomething(add) {
  console.log(add);
  const result = add(2, 3);
  console.log(result);
}
```

3. 값을 리턴하는 함수

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

// 함수 호출
doSomething();

const result = add(1, 2);
console.log(result);

doSomething(add);   // 함수 자체가 전달된다.

doSomething(add()); // NaN

doSomething(add(1, 2));
```

4.함수를 변수에 할당

```
const addFun = add;
console.log(add);
const result = addFun(1, 2);
console.log(result);
```

☑ Function

- fundamental building block in the program
- subprogram can be used multiple times
- performs a task or calculates a value

(1) Function declaration

```
/*
function name(param1, param2) {
    body ...
    return;
}

one function === one thing
naming: doSomething, command, verb

e.g)
createCardAndPoint -> createCard, createPoint

function is object in JS
*/

function printHello() {
  console.log('Hello');
}
printHello();

function log(message) {
  console.log(message);
}
log('Hello!!!');
log(1234);
```

(2) Parameters

```
// primitive parameters: pass by value
// object parameters: passwd by reference
function changeName(obj) {
  obj.name = 'ecom';
}
const smu = { name: 'smu' };
changeName(smu);
console.log(smu);
```

(3) Default parameters (added in ES6)

```
function showMessage(message, from = 'unknown') {
    if (from === undefined) {
```

```
        from = 'unknown';
    }
  console.log(`${message} by ${from}`);
}

showMessage('Hi~');
```

(4) Rest parameters (added in ES6)

```
function printAll(...args) {    // ... => 배열 형태로 전달됨.
  // 1)
  for (let i = 0; i < args.length; i++) {
    console.log(args[i]);
  }

  // 2)
  for (const arg of args) {
    console.log(arg);
  }

  // 3)
  args.forEach((arg) => console.log(arg));
}

printAll('ecom01', 'ecom', 'smu');
```

(5) Local scope

```
// 밖에서는 안이 보이지 않고, 안에서만 밖을 볼 수 있다.
let globalMessage = 'global'; //global variable
function printMessage() {
  let message = 'hello';
  console.log(message);   //local variable
  console.log(globalMessage);
  // 중첩된 함수에서 자식의 함수가 부모 함수에 정의된 변수들의
  // 접근이 가능한 것을 클로저(closure)라고 한다.
  function printAnother() {
    console.log(message);
    let childMessage = 'hello';
  }
  // return undefined; 생략 가능
}
printMessage();
//console.log(message); //error 발생
```

```
console.clear();
```

(6) Return a value

```
function sum(a, b) {
  return a + b;
}
//const result1 = sum(1, 2);   // 3
console.log(`sum: ${sum(1, 2)}`);
```

(7) Early return, early exit

```
// bad
function ungradeUser(user) {
  if (user.point > 10) {
    //long upgrade logic...
  }
}

// good
// 조건이 맞지 않을 경우, 값이 undefined 인 경우, 값이 0인 경우에는
// 빨리 return 하고 필요한 로직은 그 뒤에 작성한다.
function ungradeUser(user) {
  if (user.point <= 10) {
    return;
  }
  //long upgrade logic...
}
```

☑ Function Expression

```
// First-class function
// functions are treated like any other variable
// can be assigned as a value to variable
// can be passed as an argument to other functions.
// can be returned by another function
// 함수는 다른 변수와 마찬가지로 변수에 할당이되고,
// 함수에 인자로 전달이 되며,
// 다름 함수의 rerurn 값으로 전달이 된다.
```

(1) Function expression

```
// a function declaration can be called earlier than it is defined. (hoisted)
// function declaration은 hoisted가 된다.
// 그말은 함수가 선언되기 전에 호출하여도 가능하다.
// 이유는 JavaScript 엔진이 선언된 것을 제일 위로 올려주기 때문이다.
//
// a function expression is created when the execution reaches it.
```

```
// function expression은 할당된 다음부터 호출이 가능하며,
// 선언 전에 호출하면 에러가 발생한다.
const print = function () {   // anonymous function
  console.log('print');
};
print();

const printAgain = print;
printAgain();

const sumAgain = sum;
console.log(sumAgain(1, 3));


(2) Callback function using function expression
function randomQuiz(answer, printYes, printNo) {
  if (answer === 'love you') {
    printYes();
  } else {
    printNo();
  }
}

// anonymous function
const printYes = function () {
  console.log('yes!');
};


// named function
// better debugging in debugger's stack traces
// recursions
const printNo = function print() {
  console.log('no!');
  //print();  // recursions -> call stack error 발생
};
randomQuiz('ecom', printYes, printNo);
randomQuiz('love you', printYes, printNo);


// Arrow function
// always anonymous
// const simplePrint = function () {
//   console.log('simplePrint!');
// };

const simplePrint = () => console.log('simplePrint!');
```

```
//const add1 = (a, b) => a + b;
const simpleMutiply = (a, b) => {
  //do something more
  return a * b;
};
```

☑ IIFE: Immediately Invoked Function Expression

```
(function hello() {
  console.log('IIFE');
})();
//hello();
```

```
// Fun quiz time
// function calculate(command, a, b)
// command: add, substract, divide, multiply, remainder

function calculate(command, a, b) {
  switch (command) {
    case 'add':
      return a + b;
    case 'substract':
      return a - b;
    case 'divide':
      return a / b;
    case 'multiply':
      return a * b;
    case 'remainder':
      return a % b;
    default:
      throw Error('unknown command');
  }
}
```