01

함수의 개념과 선언 방법

학습내용

01 함수란 무엇인가?

02 함수의 정의

03 함수의 호출과 매개변수

04 함수의 반환문

05 함수 실습

학습목표

- 함수란 무엇인지 설명할 수 있다.
- 함수의 구성요소를 이해하고, 코드로 정의할 수 있다.
- 함수의 호출과 매개변수에 대해 설명할 수 있다.
- 함수의 반환문을 작성할 수 있다.
- 함수를 직접 정의하고 호출하는 실습을 통해 효율적인 프로그래밍을 할 수 있다.

지난 주차 복습

제어문

종류	기본 문법 예시	사용 목적	주요 특징
if / else	if (조건) { } else { }	조건에 따라 실행 분기	else if로 다중 조건 가능
switch / case	switch(값) { case A: break; }	여러 값 중 하나를 선택	break 생략 시 연속 실행
for	for (초기값; 조건; 증감) { }	반복 횟수가 정해졌을 때	인덱스 기반 반복
while	while (조건) { }	조건이 참일 동안 반복	조건 먼저 확인
do...while	do { } while (조건);	조건과 상관없이 한 번은 실행	조건 나중에 확인

지난 주차 복습

제어문

종류	기본 문법 예시	사용 목적	주요 특징
for...in	for (key in 객체) { }	객체의 키 순회	배열에는 권장되지 않음
for...of	for (item of 배열) { }	배열, 문자열 등 순회	이터러블 대상 전용
break	break;	반복 또는 switch문 종료	루프 또는 switch 즉시 탈출
continue	continue;	다음 반복으로 건너뛰기	현재 반복 건너뜀

생각 해보기

Q 무한 반복문은 항상 나쁠까? 유용할 때도 있을까?
유용한 경우가 있다면 어느 경우일까?
아래처럼 유용한 경우를 생각해 보고,
해당 사례를 게시판에 올려주세요.

예
이벤트 프로그램에 반복이 사용될 경우

01 프로그램을 시작한다.
02 상태를 "계속 반복"으로 설정한다.
03 반복한다.(상태가 "계속 반복"인 동안)
   1) 이벤트가 발생했는지 확인한다.
   2) 이벤트가 발생하지 않았다면 다시 반복한다.
   3) 이벤트가 발생했다면 상태를 "중지"로 바꾼다.
04 반복을 종료하고 이벤트 처리를 수행한다.
05 프로그램을 종료한다.

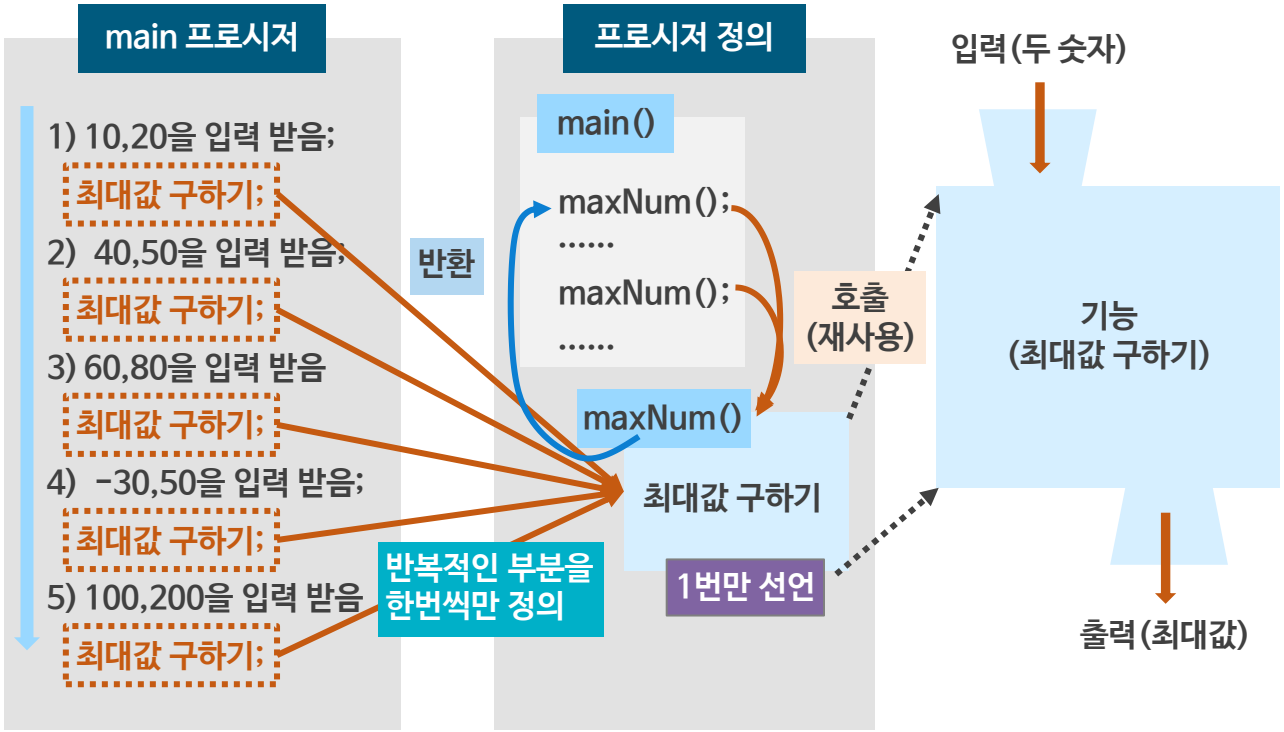
01

함수란 무엇인가?

함수

특정 작업(기능)을 수행하는 독립적인 코드 블록으로.
재사용 가능하고, 프로그램의 구조 모듈화

1번의 선언 + 여러 번의 호출로 코드를 재사용함



2) 함수의 필요성

코드의 재사용	동일한 작업을 반복적으로 수행하는 코드를 함수로 만들어 사용하는 것이 효율적임
유지보수 편의성 & 코드의 신뢰성	같은 코드의 중복으로 인한 수정에 걸리는 시간과 사람의 실수를 억제하고 재사용성을 높일 수 있음
코드의 가독성	적절한 함수 이름은 함수 내부 코드를 이해하지 않고도 함수의 역할을 파악할 수 있게 돕고, 이는 코드의 가독성을 향상시킴

참고

JavaScript의 함수는 객체 타입의 값임

02

함수의 정의

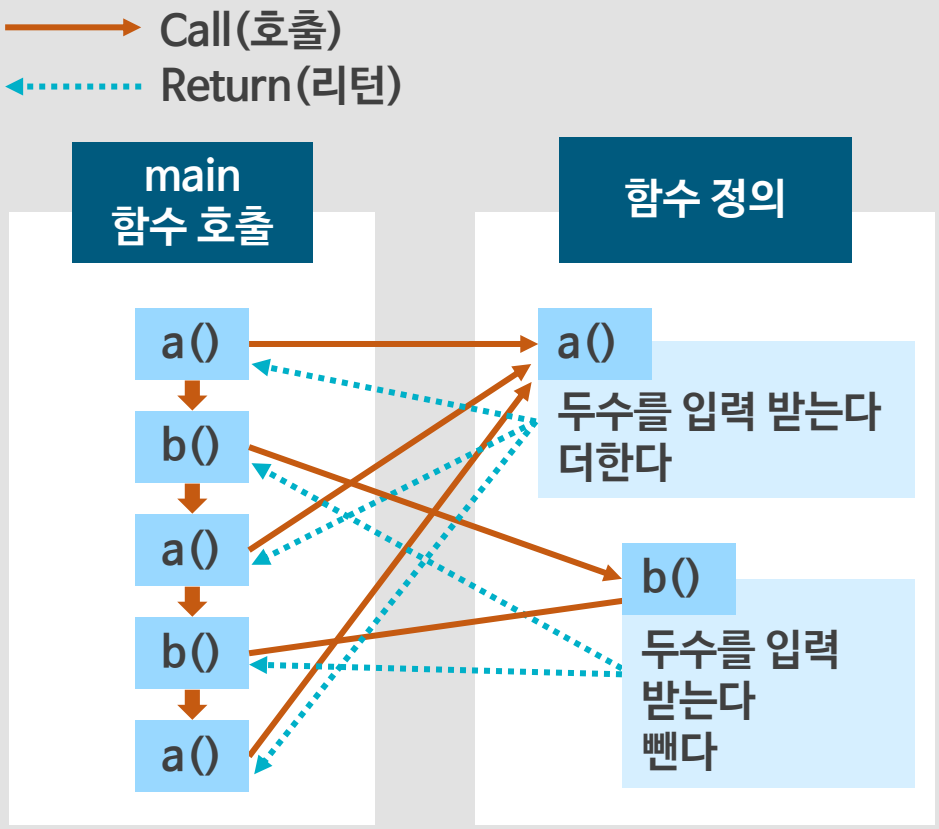
1) 함수의 구성요소

함수 = 1번의 선언 + n번의 호출

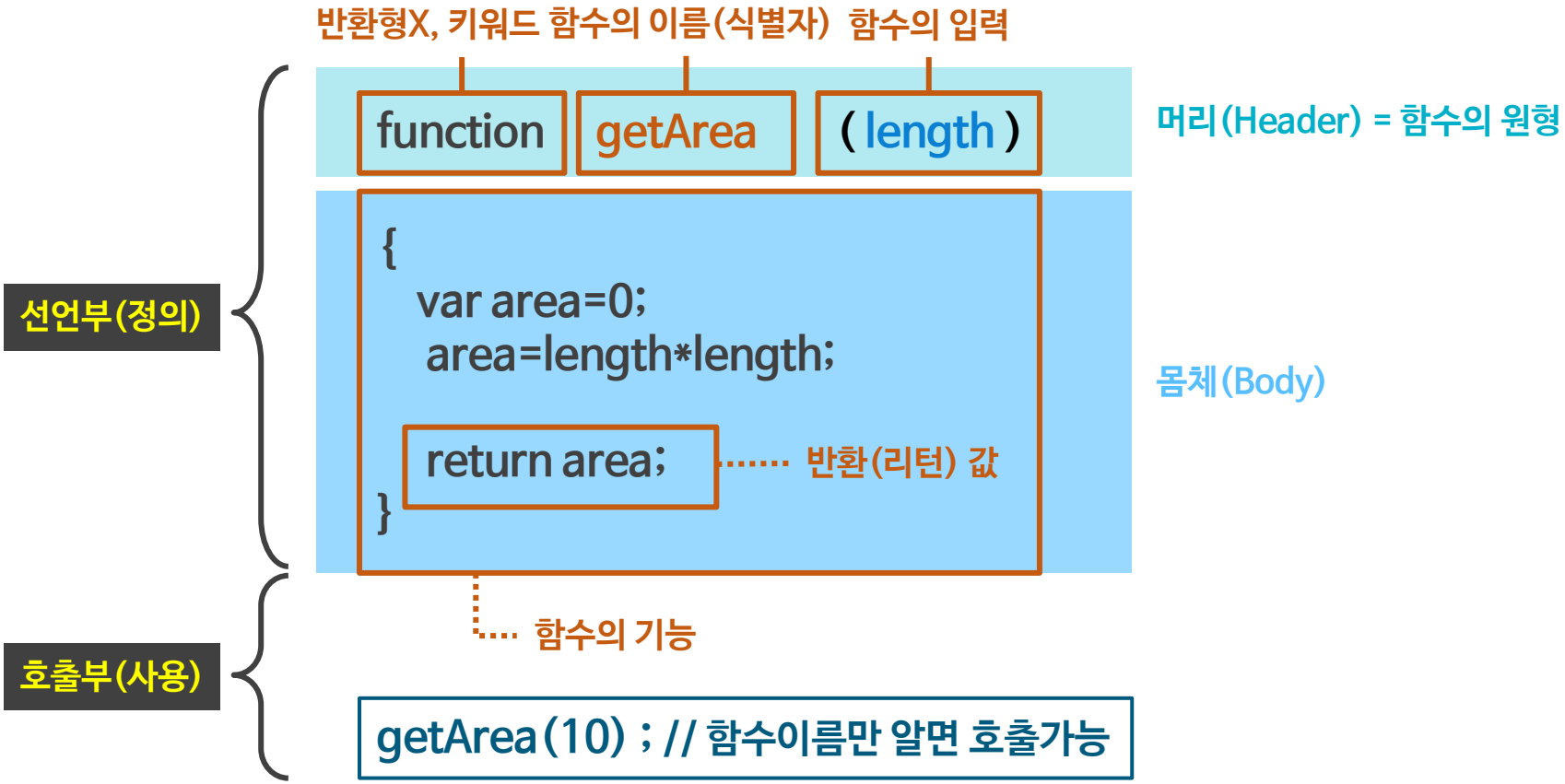
참고

프로시저 = 1번 선언 + n번 사용

- 선언부와 호출부가 분리됨
- 호출 시 내부 코드를 모르더라도 기능을 사용할 수 있음
- 함수는 반드시 호출한 곳으로 복귀(Return)해야 함



2) 함수의 정의


반환형X, 키워드 함수의 이름(식별자) 함수의 입력
function
getArea
(length)
머리(Header) = 함수의 원형
{
var area=0;
area=length*length;
return area;
}
반환(리턴) 값
몸체(Body)
선언부(정의)
함수의 기능
호출부(사용)
getArea(10) ; // 함수이름만 알면 호출가능

03

함수의 호출과 매개변수

1) 매개변수와 인수

매개변수(parameter)

- 함수를 정의할 때 사용하는 입력 값의 변수 이름
- 함수 외부에서 전달되는 값을 받는 역할 수행

인수(argument)

- 함수를 호출할 때 실제로 전달하는 값

```
//함수의 정의부
function greet(name) {
  console.log("Hello, " + name);
}


//함수의 호출부
greet("Alice");
// 출력: Hello, Alice
```

2) 매개변수의 유효범위

함수의 매개변수(parameter)는 함수 몸체 내부에서만 참조할 수 있음

매개변수의 스코프(유효 범위)는 함수 내부임

```
function add(x, y) {
  console.log(x, y); // 1 2
  return x + y;
}
add(1, 2);
console.log(x, y); // ReferenceError: x is not defined


// 매개변수의 개수 > 인수의 개수 = 나머지 매개변수 undefined
function mul(x, y) {
  console.log(x, y); // y undefined
}
mul(1); //인수
```

ReferenceError : x is not defined

3) 매개변수와 인수의 개수

함수는 매개변수(parameters) 의 개수와
인수(argument)의 개수가 일치하지 않아도 됨

인수가 매개변수보다 부족한 경우

나머지 매개변수에
대해서는 암묵적으로 undefined

인수가 매개변수보다 많은 경우

모든 인수는 암묵적으로 arguments
객체에 프로퍼티로 보관

```
// 매개변수의 개수 < 인수의 개수 = arguments 에 보관
function sub(x, y) {
  console.log(arguments); // [arguments] { '0': 3, '1': 2, '2': 1 }
  return x - y;
}
sub(3, 2, 1); // 1
```

선언 없이 arguments 를 암묵적으로 사용 가능

4) 매개변수 사용 시 고려사항

- JavaScript 함수는 매개변수와 인수의 개수가 일치하는지 확인하지 않음

- JavaScript는 '동적 타입 언어'임
  - JavaScript 함수는 매개변수의 타입을 사전에 지정할 수 없음

4) 매개변수 사용 시 고려사항

JavaScript의 경우 함수를 정의할 때,
인수가 전달되었는지 확인할 필요가 있음

01 typeof 연산자를 사용하는 방법

02 인수가 전달되지 않은 경우, 단축 평가를 사용하는 방법

03 매개변수에 기본값(Default Value)을 할당하는 방법

04 정적 타입 선언이 가능한 Typescript를 사용하는 방법

4) 매개변수 사용 시 고려사항

01 typeof 연산자를 사용하는 방법

```
// typeof 연산자로 arguments 문제 방지
function add(x, y) {
  if (typeof x !== "number" || typeof y !== "number") {
    throw new TypeError("인수는 모두 숫자(number)값 이어야 합니다.");
  }

  return x + y;
}
console.log(add(1, 2)); // 3
console.log(add(2)); // TypeError: 인수는 모두 숫자(number)값 이어야 합니다.
console.log(add("a", "b")); // TypeError: 인수는 모두 숫자(number)값 이어야 합니다.
```

typeof 연산자를 사용하는 방법

4) 매개변수 사용 시 고려사항

세종사이버대학교

02 인수가 전달되지 않은 경우, 단축 평가를 사용하는 방법

```
function mul(a, b, c) {
  a = a || 1;
  b = b || 1;
  c = c || 1;

  return a * b * c;
}
console.log(mul(1, 2, 3)); // 6
console.log(mul(1, 2)); // 2
console.log(mul(1)); // 1
console.log(mul()); // 1
```

인수가 전달되지 않은 경우, 단축 평가를 사용하는 방법

➡ 만약 a가 undefined, 0, false, null, NaN, " 중 하나라면 1로 대체되고, 이들 중 아무것도 아닐 경우에는 원래 값(a,b,c)을 그대로 출력

단축 평가 (Short-Circuit Evaluation)란?

논리 연산자 ||(OR)와 &&(AND)를 사용할 때, 앞의 조건만으로 결과가 확정되면 뒤를 평가하지 않고 넘어가는 방식

03 매개변수에 기본값(Default Value)을 할당하는 방법

```
// 매개변수 기본값 설정으로 인수 매칭 불일치 문제 방지
function sub(a = 0, b = 0) {
  return a - b;
}
console.log(sub(10, 9)); // 1
console.log(sub(10)); // 10
console.log(sub()); // 0
```

매개변수에 기본값(Default Value)을 할당하는 방법

4) 매개변수 사용 시 고려사항

04 정적 타입 선언이 가능한 Typescript를 사용하는 방법

```
// TypeScript에서는 매개변수와 반환 타입을 명시
function add(a: number, b: number ): number {
  return a + b;
}

// 사용 예
console.log(add(3, 5)); // 출력: 8
// console.log(add("3", 5)); // ✕ 오류 : string 타입은 number에 할당할 수 없음
```

정적 타입 선언이 가능한 Typescript를 사용하는 방법

5) 가변 인자(Rest Parameter)

... (나머지 인자)문법을 사용해 여러 인자를 배열로 받음

```
function printFruits(...fruits) {
  console.log(fruits);
}

printFruits("사과", "바나나", "포도");
printFruits("사과", "바나나");
```

fruits = ["사과", "바나나", "포도"]로 인식

6) arguments 객체

- 함수 호출 시 전달된 인자들을 담는 유사 배열 객체
- 객체함수 선언 방식에서만 사용 가능(function 키워드)

```
function showArgs( ) {
   // 화살표 함수(arrow function)에서는 arguments가 생성되지 않음
   console.log(arguments); // 유사 배열 객체
}
showArgs("a", "b", "c");
```

04

함수의 반환문

1) return 명령문

- return문은 함수의 실행 결과를 호출한 곳으로 반환하는 역할을 함
- return을 만나면 함수는 즉시 종료되며, 뒤에 있는 코드는 실행되지 않음
- 반환값이 없으면 undefined를 반환함

```
function sayHello() {
  return "Hello, world!";
}

const message = sayHello(); // "Hello, world!"
console.log(message);       // 출력: Hello, world!
```

실행 결과를 호출한 곳으로 반환하는 역할

- 다양한 형태의 반환 값

```
// 숫자 반환
function getNumber() {
  return 42;
}

// 객체 반환
function getUser() {
  return { name: "Alice", age: 25 };
}
```

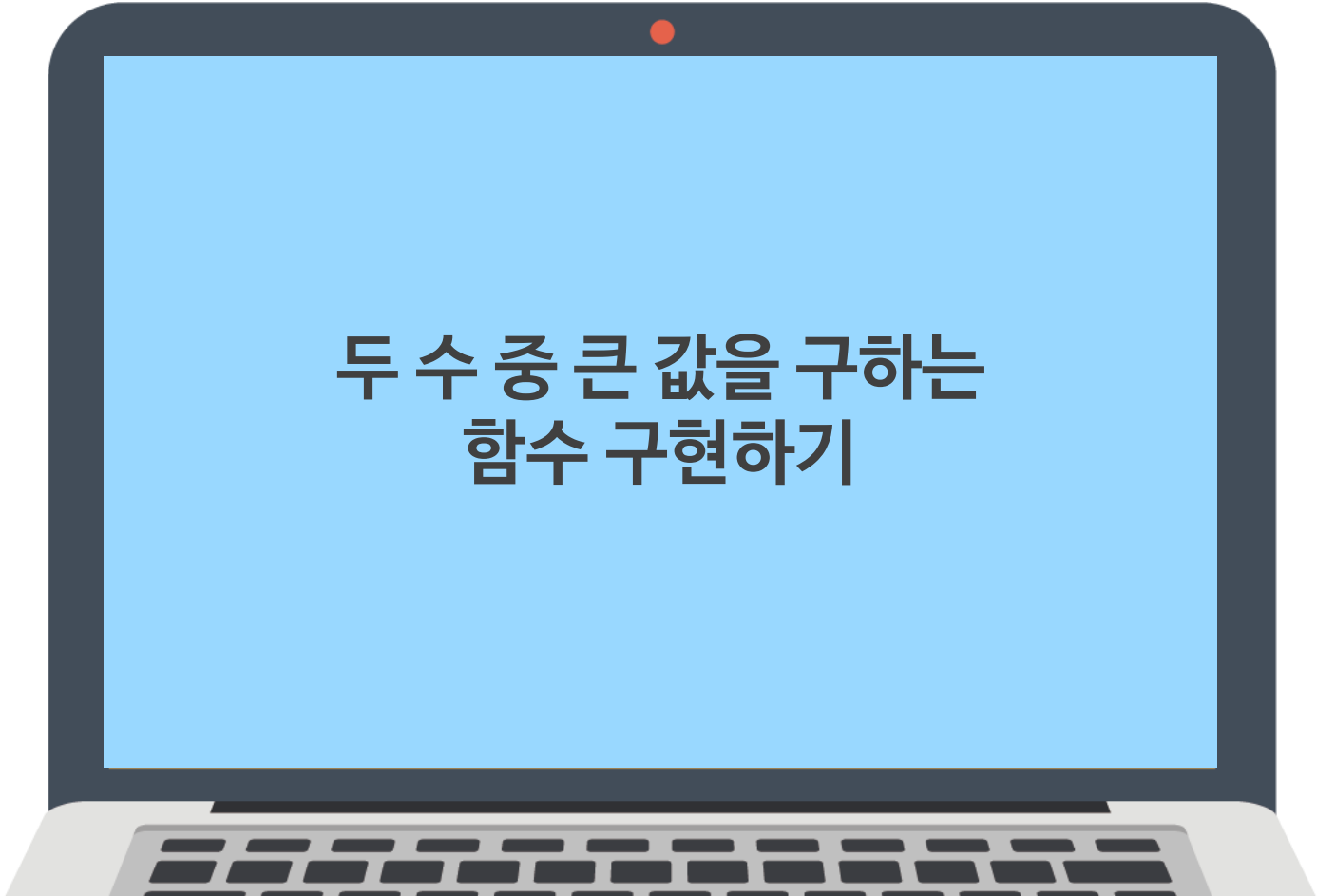
```
// 조건부 반환
function isAdult(age) {
  return age >= 18;
}
```

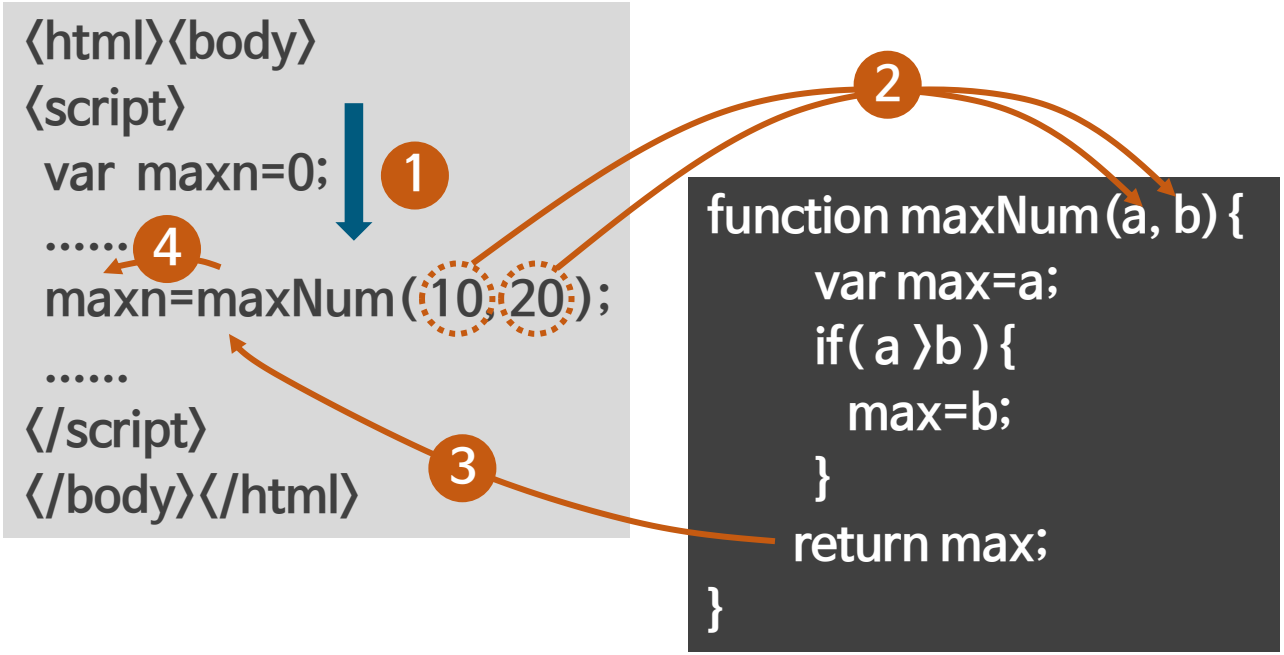
05

함수 실습

1) 용어 정리

함수에서 사용하는 용어	설명
함수(Function)	특정 작업을 수행하는 코드 묶음
매개변수(Parameter)	함수 정의 시 입력받는 변수
인자(Argument)	함수 호출 시 전달하는 실제 값
기본값(Default Parameter)	인자가 생략되었을 때 사용하는 값
가변 매개변수(Rest Parameter)	여러 개의 인자를 하나의 배열로 받는 매개변수
arguments 객체	함수 호출 시 전달된 인자들을 담는 유사 배열 객체
반환값(Return)	함수가 수행한 결과로 돌려주는 값


두 수 중 큰 값을 구하는
함수 구현하기


⟨html⟩⟨body⟩
⟨script⟩
var maxn=0;
1
.....
4
maxn=maxNum(10, 20);
.....
⟨/script⟩
⟨/body⟩⟨/html⟩
2
3
function maxNum(a, b) {
var max=a;
if(a ⟩b) {
max=b;
}
return max;
}

02

값과 리터럴