SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

ECMAScript 6
함수

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

함수 활용법

함수 in JavaScript

- 참조 타입 중 하나로써 function 타입에 속함
- JavaScript에서 함수를 정의하는 방법은 주로 2가지로 구분
  - 함수 선언식 (function declaration)
  - 함수 표현식 (function expression)
- (참고) JavaScript의 함수는 일급 객체*(First-class citizen)에 해당
  - 일급 객체*: 다음의 조건들을 만족하는 객체를 의미함
    - 변수에 할당 가능
    - 함수의 매개변수로 전달 가능
    - 함수의 반환 값으로 사용 가능

함수 선언식(function statement, declaration)

```
function name(args) {
  // do something
}
```

```
function add(numOne, numTwo) {
  return numOne + numTwo
}

const result = add(1, 2)
console.log(result) // 3
```

- 함수 선언식
  - 함수의 이름과 함께 정의하는 방식
  - 3가지 부분으로 구성
    - 함수의 이름 (name)
    - 매개변수 (args)
    - 몸통 (중괄호 내부)

함수 표현식(function expression)

```
const myFunction = function (args) {
  // do something
}
```

```
const add = function (numOne, numTwo) {
  return numOne + numTwo
}

const result = add(1, 2)
console.log(result) // 3
```

- 함수 표현식
  - 함수를 표현식* 내에서 정의하는 방식
    - (참고) 표현식*: 어떤 하나의 값으로 결정되는 코드의 단위
  - 함수의 이름을 생략하고 익명 함수*로 정의 가능
    - 익명 함수*(anonymous function): 이름이 없는 함수
    - 익명 함수는 함수 표현식에서만 가능
  - 3가지 부분으로 구성
    - 함수의 이름 (생략 가능)
    - 매개변수 (args)
    - 몸통 (중괄호 내부)

기본 인자(default arguments)

- 인자 작성 시 '=' 문자 뒤 기본 인자 선언 가능

```
const greeting = function (name = 'noName') {
    console.log(`hi ${name}`)
}

greeting() // hi noName
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

선언식 vs 표현식

함수 선언식과 표현식 비교 정리

	함수 선언식 (declaration)	함수 표현식 (expression)
공통점	**데이터 타입**, 함수 구성 요소 (이름, 매개변수, 몸통)	
차이점	익명 함수 불가능 호이스팅 O	익명 함수 가능 호이스팅 X
비고		Airbnb Style Guide 권장 방식

함수의 타입

- 선언식 함수와 표현식 함수 모두 타입은 function으로 동일

```
// 함수 표현식
const add = function (args) { }

// 함수 선언식
function sub(args) { }

console.log(typeof add) // function
console.log(typeof sub) // function
```

호이스팅(hoisting) – 함수 선언식

- 함수 선언식으로 선언한 함수는 var로 정의한 변수처럼 hoisting 발생
- 함수 호출 이후에 선언 해도 동작

```
add(2, 7) // 9

function add (num1, num2) {
    return num1 + num2
}
```

호이스팅(hoisting) – 함수 표현식

- 반면 **함수 표현식**으로 선언한 함수는 **함수 정의 전에 호출 시 에러 발생**
- 함수 표현식으로 정의된 함수는 **변수로 평가되어 변수의 scope 규칙을 따름**

```
sub(7, 2) // Uncaught ReferenceError: Cannot access 'sub'
before initialization

const sub = function (num1, num2) {
    return num1 - num2
}
```

(참고) 호이스팅(hoisting) – 함수 표현식

- 함수 표현식을 var 키워드로 작성한 경우,

  변수가 선언 전 **undefined 로 초기화 되어 다른 에러가 발생**

```
console.log(sub) // undefined
sub(7, 2) // Uncaught TypeError: sub is not a function

var sub = function (num1, num2) {
    return num1 - num2
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Arrow Function

화살표 함수 (Arrow Function)

- 함수를 비교적 간결하게 정의할 수 있는 문법
- **function 키워드 생략 가능**
- 함수의 **매개변수가 단 하나 뿐이라면, '( )' 도 생략 가능**
- 함수 **몸통이 표현식 하나라면 '{ }'과 return도 생략 가능**

Arrow Function

```
const arrow = function (name) {
  return `hello! ${name}`
}

//1. function 키워드 삭제
const arrow = (name) => { return `hello! ${name}` }


//2. () 생략 (함수 매개변수가 하나일 경우만)
const arrow = name => { return `hello! ${name}` }


//3. {} & return 생략 (바디가 표현식 1개인 경우만)
const arrow = name => `hello! ${name}`
```

함수 실습

- 목표: JavaScript 함수 연습 (05-functions.js)
- 문제: 파일에 작성된 주석 참고

Q1. 함수에 대한 typeof 연산자의 결과는 object이다. T/F

Q2. 함수 선언식과 표현식 모두 익명 함수로 선언이 가능하다. T/F

Q3. 화살표 함수는 바디가 한 줄이라면 중괄호와 return 구문이 생략 가능하다. T/F

함수 (Functions) Quiz

함수 (Functions) Quiz

Q1. 함수에 대한 typeof 연산자의 결과는 object이다. F

A1. 함수에 대한 typeof 연산자의 결과는 function이다.

Q2. 함수 선언식과 표현식 모두 익명 함수로 선언이 가능하다. F

A2. 함수 선언식은 함수의 이름을 반드시 지정해야 하므로 익명 함수 선언이 불가능하다.

Q3. 화살표 함수는 바디가 한 줄이라면 중괄호와 return 구문이 생략 가능하다. T