Chap 18 함수와 일급 객체

18-1 일급 객체

▼ 일급 객체 조건

- 무명의 리터럴로 생성 가능. 즉 런터임에 생성 가능
- 변수나 자료구조(객체, 배열)에 저장 가능
- 함수의 매개변수에 전달 가능
- 함수의 반환값으로 사용 가능

▼ 런타임?

컴파일타임 vs 런타임

- **컴파일타임**: 프로그램의 소스 코드를 기계어로 번역하는 시간. 주로 컴파일러 언어에서 소스 코드가 실행 파일로 변환되는 시점을 의미합니다. 이 시점에서는 변수, 함수, 객체 등의 구조가 결정됩니다.
- **런타임**: 프로그램이 실제로 실행되는 시간. 사용자가 프로그램을 실행하는 시점을 말하며, 이 때 프로그램의 논리가 실행되고 변수, 함수, 객체 등이 실제로 동작합니다.

"런타임에 생성 가능"의 의미

프로그래밍에서 일급 객체는 런타임에 생성될 수 있어야 합니다. 이는 프로그램이 실행되는 동안에 객체가 동적으로 생성될 수 있다는 의미입니다. 자바스크립트에서 함수는 일급 객체로 취급됩니다.

▼ 함수는 일급 객체인가?

정답은 ⭕

위 조건 모두 충족

```
// 1. 함수는 무명의 리터럴로 생성할 수 있다.
// 2. 함수는 변수에 저장할 수 있다.
// 런타임(할당 단계)에 함수 리터럴이 평가되어 함수 객체가 생성되고 변수
const increase = function (num) {
  return ++num;
};

const decrease = function (num) {
  return --num;
};

// 2. 함수는 객체에 저장할 수 있다.
const auxs = { increase, decrease };

// 3. 함수의 매개변수에게 전달할 수 있다.
// 4. 함수의 반환값으로 사용할 수 있다.
function makeCounter(aux) {
  let num = 0;

  return function () {
    num = aux(num);
    return num;
  };
}

// 3. 함수는 매개변수에게 함수를 전달할 수 있다.
const increaser = makeCounter(auxs.increase);
console.log(increaser()); // 1
console.log(increaser()); // 2

// 3. 함수는 매개변수에게 함수를 전달할 수 있다.
const decreaser = makeCounter(auxs.decrease);
console.log(decreaser()); // -1
console.log(decreaser()); // -2
```

▼ 일급 객체로서의 함수 특징

일급 객체라는 것은 함수를 객체와 동일하게 사용할 수 있다는 의미! 따라서 함수는 값을 사용할 수 있는 곳(변수 할당문, 객체의 프로퍼티 값, 배열의 요소, 함수 호출의 인수, 함수 반환문)이라면 어디서든지 리터럴로 정의할 수 있으며 런타임에 함수 객체로 생성된다.

특징

- 일반 객체와 같이 함수의 매개변수에 전달할 수 있다.
- 함수의 반환값으로 사용 가능.
- 즉 함수형 프로그래밍 가능케 한다.

18-2 함수 객체의 프로퍼티

▼ 함수 객체 내부확인 방법 (콘솔)

console.dir()를 통해서 콘솔에 함수 객체 내부를 출력해준다

▼ 프로퍼티 어튜리뷰트 확인 방법

Object.getOwnPropertyDescriptors메소드를 이용하여 콘솔로 찍으면 프로퍼티어튜리뷰트 확인할 수 있다.

```
function square(number) {
  return number * number;
}

console.log(Object.getOwnPropertyDescriptors(square));
/*
{
  length: {value: 1, writable: false, enumerable: false, c
  name: {value: "square", writable: false, enumerable: fal
  arguments: {value: null, writable: false, enumerable: fa
  caller: {value: null, writable: false, enumerable: false
  prototype: {value: {...}, writable: true, enumerable: fa
}
*/

// __proto__는 square 함수의 프로퍼티가 아니다.
console.log(Object.getOwnPropertyDescriptor(square, '__pro
```

```
// __proto__는 Object.prototype 객체의 접근자 프로퍼티다.
// square 함수는 Object.prototype 객체로부터 __proto__ 접근자 프
console.log(Object.getOwnPropertyDescriptor(Object.prototy
// {get: f, set: f, enumerable: false, configurable: true}
```

▼ 함수 고유의 프로퍼티 종류

- arguments
- caller
- length
- name
- prototype

위 프로퍼티는 일반 객체에는 없는 함수 고유의 데이터 프로퍼티이다.

▼ __proto__

__proto__는 접근자 프로퍼티이며 함수 고유의 프로퍼티가 아닌 Object.prototype 객체의 프로퍼티를 상속 받는다. 따라서 모든 객체가 사용 가능하다

▼ arguments

arguments 프로퍼티(속성) 값은 arguments 객체이다.

▼ arguments 객체

함수 호출시 인수들의 정보를 담고있는 순회 가능한 유사 배열 객체이며, 함수 내부에서 지역 변수처럼 작동한다. 즉, 함수 외부에서 참조 불가능

```
function multiply(x, y) {
  console.log(arguments);//함수 내부에서 참조 가능
  return x * y;
}

console.log(multiply());         // NaN
console.log(multiply(1));        // NaN
console.log(multiply(1, 2));     // 2
console.log(multiply(1, 2, 3)); // 2
```

▼ arguments 객체의 프로퍼티

```
function multiply(x, y) {
  console.log(arguments);//함수 내부에서 참조 가능
  return x * y;
}

console.log(multiply());         // NaN
console.log(multiply(1));        // NaN
console.log(multiply(1, 2));     // 2
console.log(multiply(1, 2, 3)); // 2
```

위와같은 코드일때 아래와 같은 결과가 나온다.

위사진에서 볼수 있듯이 arguments 객체의 값은 인수의 정보를 담고 프로퍼티는 아래 3가지가 있다.

- callee : arguments를 생성한 함수. 자기 자신을 가리킴.
- length : 인수의 개수
- Symbol

  ▼ Symbol

  arguments 객체를 순회가능한 자료구조인 이터러블로 만들기 위한 프로퍼티이다.

  Symbol.iterator 프로퍼티 키로 사용한 메서드로 인해 이터러블이 된다.

▼ argumebts 객체의 유용한점

자바스크립트는 매개변수와 인수의 개수가 일치하는지 확인하지 않는다. 따라서 함수 호출시 매개변수 개수만큼 인수를 전달하지 않아도 에러가 발생하지 않는다. 따라서 인수의 개수를 확인할 있는 argumenta객체는 가변 인자 함수를 구현 시에 유용하다.

▼ arguments 객체의 문제점

유사 배열이기때문에 배열메서드를 사용할 때 에러가 발생하여

Function.prototype.call, Function.prototype.apply를 사용해 간접 호출해야 하는 번거로움이 있다.

```
function sum() {  // arguments 객체를 배열로 변환
 const array = Array.prototype.slice.call(arguments);
 return array.reduce(function (pre, cur) {
  return pre + cur;  }, 0);
}
 console.log(sum(1, 2));          // 3
 console.log(sum(1, 2, 3, 4, 5)); // 15
```

▼ 해결방법

ES6에서 도입된 Rest 파라미터

```
// ES6 Rest parameter
function sum(...args) {
 return args.reduce((pre, cur) => pre + cur, 0);
}
console.log(sum(1, 2));          // 3console.log(sum
```

▼ caller

함수 자신을 호출한 함수를 가리킨다

▼ length

함수를 정의할 때 선언한 매개변수의 개수를 가리킨다

▼ name

함수 이름을 가리킨다. ES6부터 표준으로 바뀜.

ES5는 익명 함수 표현식일 경우 name 프로퍼티는 빈 문자열을 값으로 갖는다. 그러나 ES6에서는 함수 객체를 가리키는 식별자를 값으로 갖는다.

▼ prototype

prototype프로퍼티는 생성자 함수로 호출할 수 있는 함수 객체, 즉 constructor만이 소유하는 프로퍼티다.
일반 객체와 생성자 함수로 호출할 수 없는 non-constructor에는 prototype프로퍼티가 없다

▼ hasOwnProperty메소드

인수로 전달받은 키가 객체 고유의 프로퍼티키일때만 true를 반환하고 상속받은 프로퍼티일 경우에는 false를 반환한다