CHAPTER 05 함수

자바스크립트 프로그래밍 입문 (2판)

Contents

학습목표

- 함수의 기본 형태를 이해합니다.
- 함수의 매개 변수와 반환을 이해합니다.
- 함수를 변수에 저장할 수 있다는 것과 콜백 함수 사용 방법을 이해합니다.
- 자바스크립트의 표준 내장 함수를 이해합니다.

Contents

내용

- 함수 생성 방법
- 함수의 기본 형태
- 함수의 기본 활용 형태
- 함수 매개 변수 초기화
- 콜백 함수
- 표준 내장 함수
- 조금 더 나아가기

1. 함수 생성 방법

익명 함수

- 이름을 붙이지 않고 함수 생성
- 함수를 호출하면 함수 내부의 코드 덩어리가 모두 실행

```
let 함수_이름 = function () { };
```

코드 5-1 익명 함수

```
let 함수 = function () {
    console.log("함수의 첫 번째 줄");
    console.log("함수의 두 번째 줄");
};

함수();
console.log(함수);
```

함수를 생성합니다.

함수를 호출합니다.

함수 자체를 출력합니다.

실행 결과

```
함수의 첫 번째 줄
함수의 두 번째 줄
[Function: 함수]
```

1. 함수 생성 방법

선언적 함수

- 이름을 붙여 함수를 생성

```
function 함수_이름() { }
```

코드 5-2 선언적 함수

```
function 함수() {
    console.log("함수의 첫 번째 줄");
    console.log("함수의 두 번째 줄");
}

함수();
console.log(함수);
```

함수를 생성합니다.

함수를 호출합니다.

함수를 출력합니다.

실행 결과

```
함수의 첫 번째 줄
함수의 두 번째 줄
[Function: 함수]
```

- 'console.log (함수)' 부분으로 '[Function: 함수]' 문자를 출력

1. 함수 생성 방법

화살표 함수[ECMAScript6]

```
() => { }
```

- '하나의 표현식을 리턴하는 함수'를 만들 때는 중괄호 생략 가능
- 익명 함수 예제를 화살표 함수로 바꾸기

코드 5-3 화살표 함수 arrowFunction.js

```
let 함수 = () => {
    console.log("함수의 첫 번째 줄");
    console.log("함수의 두 번째 줄");
};

함수();
console.log(함수);
```

함수를 생성합니다.

함수를 호출합니다.

실행 결과

```
함수의 첫 번째 줄
함수의 두 번째 줄
[Function: 함수]
```

2. 함수의 기본 형태

- 기본 형태

```
function 함수_이름(매개_변수) {
    함수_코드;
    return 리턴_값;
}
```

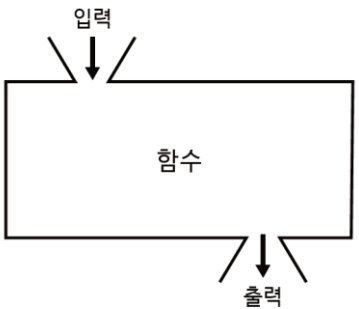


그림 5-1 함수 상자

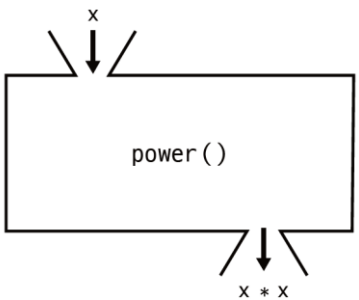


그림 5-2 power() 메소드

2. 함수의 기본 형태

- [예제 5-1] 함수의 기본 형태
  - 매개 변수로 넣은 숫자를 제곱하는 power( ) 함수 생성

코드 5-4 함수의 기본 형태 power.js

```
function power(x) {
    return x * x;
}

console.log(power(10));
console.log(power(20));
```

실행 결과

```
100
400
```

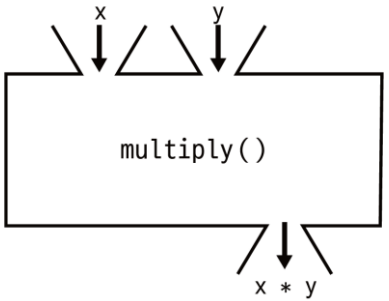
2. 함수의 기본 형태

- 매개 변수가 여러 개인 함수

코드 5-5 매개 변수가 여러 개인 함수

```
function multiply(x, y) {
    return x * y;
}

console.log(multiply(52, 273));
console.log(multiply(103, 32));
```

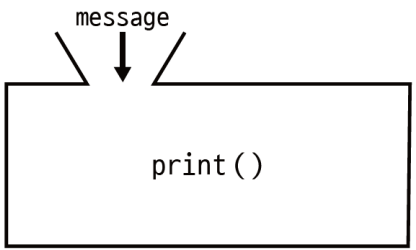


- 리턴 없는 함수

코드 5-6 리턴 없는 함수

```
function print(message) {
    console.log(`"${message}"(이)라고 말했습니다!`);
}

print("안녕하세요");
print("자바스크립트 공부");
```



3. 함수의 기본 활용 형태

- 리턴하는 함수의 기본 형태

```
function (매개_변수, 매개_변수) {
    let output = 초깃값;
    // output 계산
    return output;
}
```

- [예제 5-2] 매개 변수와 리턴(1) - min부터 max까지 숫자를 더해 리턴

코드 5-8 매개 변수와 리턴 (1) functionBasicA.js

```
function sum(min, max) {
    let output = 0;
    for (let i = min; i <= max; i++) {
        output += i;
    }
    return output;
}

console.log(sum(1, 100));
```

실행 결과

```
5050
```

3. 함수의 기본 활용 형태

- [예제 5-3] 매개 변수와 리턴(2)
  - min부터 max까지 숫자를 곱해 리턴하는 함수를 생성 호출

코드 5-9 매개 변수와 리턴 (2) functionBasicB.js

```
function multiply(min, max) {
    let output = 1;
    for (let i = min; i <= max; i++) {
        output *= i;
    }
    return output;
}

console.log(multiply(1, 10));
```

실행 결과

```
3628800
```

4. 함수 매개 변수 초기화

- 매개 변수를 입력하지 않고 함수 호출
  - 실행하면 undefined가 출력

코드 5-10 매개 변수를 입력하지 않고 함수 호출

```
// 함수를 선언합니다.
function print(name, count) {
    console.log(`${name}이/가 ${count}개 있습니다.`)
}

// 함수를 호출합니다.
print("사과", 10);
print("사과");
```

4. 함수 매개 변수 초기화

- [예제 5-4] 조건문을 활용한 매개 변수 초기화
  - 조건문으로 매개 변수를 확인, count가 undefined일 때 1로 초기화

코드 5-11 조건문을 활용한 매개 변수 초기화 — initWithCondition.js

```
// 함수를 선언합니다.
function print(name, count) {
    // 함수 매개 변수 초기화
    if (typeof(count) == "undefined") {
        count = 1;
    }

    // 함수 본문
    console.log(`${name}이/가 ${count}개 있습니다.`)
}

// 함수를 호출합니다.
print("사과");
```

실행 결과

```
사과이/가 1개 있습니다.
```

4. 함수 매개 변수 초기화

- [예제 5-5] 짧은 초기화 조건문을 활용한 매개 변수 초기화
  - 짧은 초기화 조건문으로 매개 변수를 확인, count가 undefined일 때 1로 초기화

코드 5-12 디폴트 매개 변수를 활용한 매개 변수 초기화

```
// 함수를 선언합니다.
function print(name, count = 1) {
    console.log(`${name}이/가 ${count}개 있습니다.`);
}

// 함수를 호출합니다.
print("사과");
```

실행 결과

```
사과이/가 1개 있습니다.
```

5. 콜백 함수

- 함수의 매개 변수로 전달되는 함수
- [예제 5-6] 콜백 함수 사용

코드 5-14 콜백 함수 callback.js

```
// 함수를 선언합니다.
function callTenTimes(callback) {
    // 10회 반복합니다.
    for (let i = 0; i < 10; i++) {
        // 매개 변수로 전달된 함수를 호출합니다.
        callback();
    }
}

// 변수를 선언합니다.
callTenTimes(function () {
    console.log('함수 호출');
});
```

실행 결과

```
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
함수 호출
```

6. 표준 내장 함수

- 자바스크립트에서 기본적으로 지원하는 함수

숫자 변환 함수

표 5-1 숫자 변환 함수

함수	설명
parseInt()	문자열을 정수로 변환합니다.
parseFloat()	문자열을 실수로 변환합니다.

6. 표준 내장 함수

- [예제 5-7] parseInt( ) 함수와 parseFloat( ) 함수
  - 문자열을 숫자로 변환

표 5-2 숫자 생성 방법

숫자 생성 방법	설명
0숫자	8진수 숫자를 만듭니다.
숫자	10진수 숫자를 만듭니다.
0x숫자	16진수 숫자를 만듭니다.

코드 5-15 parseInt() 함수와 parseFloat() 함수 parseNumber.js

```
// 변수를 선언합니다.
let inputA = "52";
let inputB = "52.273";
let inputC = "1401동"

// parseInt() 함수의 기본적인 사용
console.log(parseInt(inputA))

// parseInt() 함수와 parseFloat() 함수의 차이
console.log(parseInt(inputB))
console.log(parseFloat(inputB))

// 문자열 뒤에 숫자가 아닌 문자가 포함되어 있을 때
console.log(parseInt(inputC));
```

실행 결과

```
52
52
52.273
1401
```

6. 표준 내장 함수

타이머 함수

- '특정 시간 후에' or '특정 시간마다' 어떤 일을 할 때 사용
- 시간은 밀리초로 지정. 1초를 나타내려면 1000(밀리초)을 입력

표 5-3 타이머 설정 함수

함수	설명
setTimeout(함수, 시간)	특정 시간 후에 함수를 실행합니다.
setInterval(함수, 시간)	특정 시간마다 함수를 실행합니다.

6. 표준 내장 함수

- [예제 5-8] 타이머 함수
  - 1초 후에 '1초가 지났습니다.', 1초마다 '1초 마다 호출됩니다.'를 출력

코드 5-16 타이머 함수 timer.js

```
// 1초 후에
setTimeout(function () {
    console.log("1초가 지났습니다.");
}, 1000);

// 1초마다
setInterval(function () {
    console.log("1초 마다 호출됩니다.");
}, 1000);
```

실행 결과

```
1초가 지났습니다.
1초 마다 호출됩니다.
1초 마다 호출됩니다.
1초 마다 호출됩니다.
^C
```

표 5-4 타이머 제거 함수

함수	설명
clearInterval(아이디)	특정 시간마다 실행하던 함수 호출을 정지합니다.

  - 종료 : Ctrl + C
  - 실행 화면에서 ^C는 Ctrl + C 를 눌렀다는 의미임

6. 표준 내장 함수

- [예제 5-9] clearInterval( ) 함수

코드 5-17 clearInterval() 함수 clearTimer.js

```
// 1초마다
let id = setInterval(function () {
    console.log("출력합니다.");
}, 1000);

// 3초 후에
setTimeout(function () {
    // 타이머를 제거합니다.
    clearInterval(id);
}, 3000);
```

실행 결과

```
출력합니다.
출력합니다.
```

7. 조금 더 나아가기

익명 함수와 선언적 함수의 생성 순서

코드 5-18 변수 덮어쓰기

```
let 변수;
변수 = 10;
변수 = 20;

console.log(변수)
```

기존의 값인 10 대신 20으로 덮어씀

실행 결과

```
20
```

코드 5-19 함수 덮어쓰기 (1)

```
let 함수;
함수 = function () { console.log("첫 번째 함수"); };
함수 = function () { console.log("두 번째 함수"); };

함수();
```

실행 결과

```
두 번째 함수
```

7. 조금 더 나아가기

코드 5-20 함수 덮어쓰기 (2)

```
함수 = function () { console.log("첫 번째 함수"); };
function 함수() { console.log("두 번째 함수"); };

함수();
```

실행 결과

```
첫 번째 함수
```

예제 A

```
함수 = function () { console.log("1"); };
함수 = function () { console.log("2"); };

함수();
```

예제 B

```
함수 = function () { console.log("1"); };
function 함수() { console.log("2"); };

함수();
```

예제 C

```
function 함수() { console.log("1"); };
함수 = function () { console.log("2"); };

함수();
```

예제 D

```
function 함수() { console.log("1"); };
function 함수() { console.log("2"); };

함수();
```

7. 조금 더 나아가기

익명 함수와 화살표 함수의 차이

- 내부에서 this 키워드가 가지는 의미

코드 5-21 익명 함수와 화살표 함수의 차이

```
// 익명 함수 생성 후 곧바로 호출
(function () {
    console.log(this);
})();
```

```
// 화살표 함수 생성 후 곧바로 호출
(() => {
    console.log(this);
})();
```

실행 결과

```
<ref *1> Object [global] {
  global: [Circular *1],
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  },
  queueMicrotask: [Function: queueMicrotask],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Getter]
  }
}
```

실행 결과

```
{}
```


THANK
YOU!