함수

함수

- **함수(Function)**

  특정 동작을 수행하거나 코드를 재사용하기 위해 사용

- **함수의 종류**

  1. 일반 함수
  2. 익명 함수
  3. 화살표 함수
  4. 생성자 함수
  5. 내부 함수
  6. 재귀 함수
  7. 콜백 함수
  8. ...

함수의 종류 - 1

- 일반 함수

일반적으로 사용되는 함수로써 return문을 사용하여 값을 반환함
* 호이스팅 가능

```
// 일반 함수
function add(a, b) {
    return a + b;
}

console.log(add(5,10));
```

함수의 종류 - 2

- **익명 함수**

익명 함수는 일반적으로 한번만 사용하는 기능이 필요할 경우에 쓰임

→ 메모리 관리에 대한 방안으로, 일반 함수는 호이스팅 되어 불필요한 메모리를 차지하게 되지만 익명 함수는 실행 후 사라짐

```
// 익명 함수
var add = function(a, b) {
    console.log(a + b);
};

add(3,8);
```

함수의 종류 - 3

- **화살표 함수**

코드의 라인을 줄이기 위해 사용

```
// 화살표 함수
const add = (a, b) => a+b;
console.log(add(1,7));
```

함수의 종류 - 4

- **생성자 함수**

객체를 생성하여 재사용 하기 위해 사용됨
→ 코드의 재사용이 아닌, 객체의 재사용
→ 관례적으로 생성자 함수는 앞 글자를 대문자로 표시함

```
// 생성자 함수
function Parson(name, age) {
    this.name = name;
    this.age = age;
    this.walk = function() {
        console.log("걷는중");
    }
}

const p1 = new Parson("김재섭", 19);
const p2 = new Parson("홍길동", 20);

console.log(p1);
console.log(p2);

console.log(p1.name);
console.log(p1.walk());
```

함수의 종류 - 5

- **생성자 함수에서 undefined 발생 이유**

p1.walk()를 실행할 때 console.log 코드는 실행이 되었지만, 함수가 종료되며
반환 값이 없기 때문에 undefined 발생

```
// 생성자 함수
function Parson(name, age) {
    this.name = name;
    this.age = age;
    this.walk = function() {
        // console.log("걷는중");
        return "걷는중";
    }
}

const p1 = new Parson("김재섭", 19);
const p2 = new Parson("홍길동", 20);

console.log(p1);
console.log(p2);

console.log(p1.name);
console.log(p1.walk());
```

함수의 종류 - 6

- 내부 함수와 재귀 함수

내부 함수 : 함수 내에서 또 다시 정의된 함수
재귀 함수 : 함수가 자기 자신을 또 다시 호출하는 함수

```
// 내부 함수
function outer() {
    let x = 10;

    function inner() {
        console.log(x);
    }

    inner();
}

outer();
```

```
// 재귀함수
function countdown(cnt) {
    if (cnt === 0) {
        console.log("0이 되었습니다");
    } else {
        console.log(cnt);
        countdown(cnt - 1 );
    }
}

countdown(5);
```

함수의 종류 - 7

- **콜백 함수**

함수의 파라미터에 함수가 들어가며, 순차적으로 실행하고 싶을 때 사용
→ 비동기 프로그래밍을 할 때 사용됨

```
myButton.addEventListener('click', function(event) {
    myButton.textContent = '클릭되었습니다!';
    output.textContent = '버튼이 클릭되었습니다.';
});
```

```
setTimeout(function() {
    console.log("Timeout");
}, 5000)
```

함수의 종류 - 8

- **콜백 함수 사용 예시**

아래와 같이 다양한 형태로 사용이 가능 하지만, 만약 함수 호출은 그대로인데
팀원마다 출력되기 원하는 내용이 다를 경우 왼쪽 이미지와 같이 사용하여
함수의 일부 기능만을 수정하여 순차적으로 실행 및 사용이 가능

```
function first(callback) {
    console.log("First");
    if(typeof callback === 'function') {
        callback();
    }
}

first(function() {
    console.log("Second");
})
```

```
function first(callback) {
    console.log("First");
    callback();
}

function second() {
    console.log("Second");
}

first(second);
```

함수의 종류 - 9

- **콜백 지옥**

콜백 함수가 반복되어 코드의 들여쓰기가 감당하기 힘들정도로 깊어지는 현상
Promise 또는 async await을 사용하여 콜백 지옥을 방지 할 수 있음

```
first(function() {
    second(function() {
        third(function() {
            fourth(function() {
                fifth(function() {
                    sixth(function() {
                        seventh(function() {
                            eighth(function() {
                                // 콜백 지옥의 끝
                            });
                        });
                    });
                });
            });
        });
    });
});
```