제너레이터

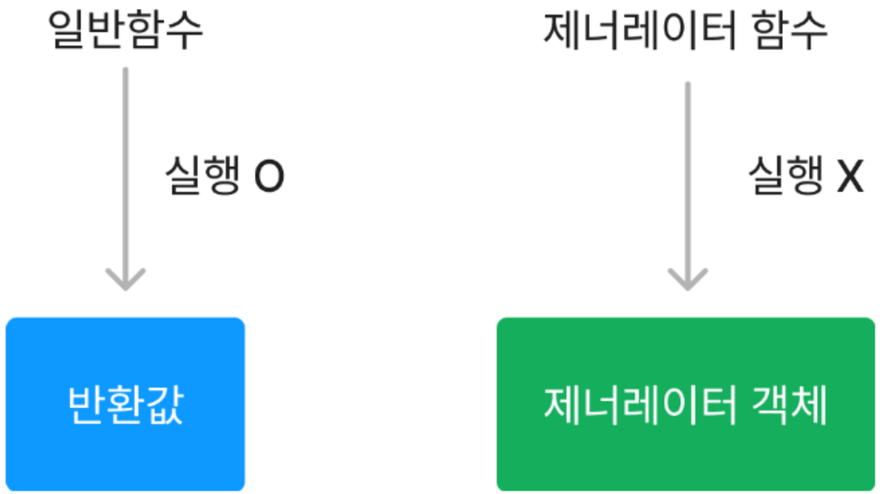
이은주

2023.06.03

제너레이터

- 함수의 실행을 중간에 멈췄다가 재개할 수 있는 특별한 함수

제너레이터



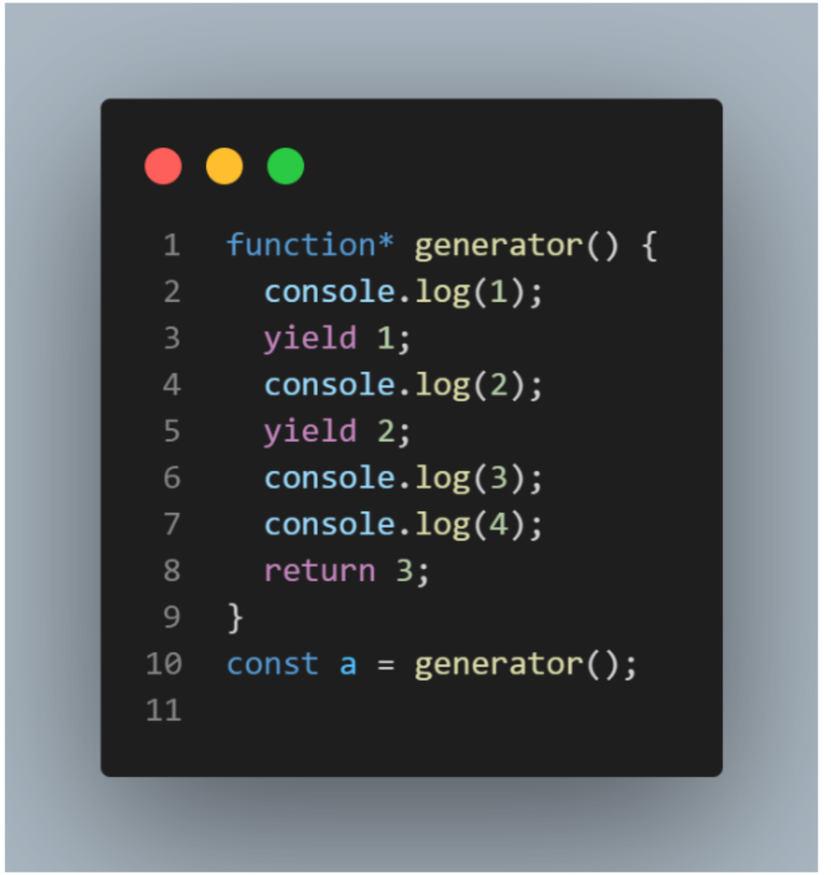

```
function normal() {
    console.log(1);
}
undefined
normal()
1

> function* generator() {
      console.log(1);
  }
< undefined
> generator()
< ▶ generator {<suspended>}
```

제너레이터

yield 키워드

- 제너레이터 함수에서 값을 반환하면서 실행을 일시 중단
- 함수가 다시 호출될 때 일시 중단된 지점에서 실행을 재개하며, 이전 상태를 유지



```
1  function* generator() {
2    console.log(1);
3    yield 1;
4    console.log(2);
5    yield 2;
6    console.log(3);
7    console.log(4);
8    return 3;
9  }
10 const a = generator();
11
```

제너레이터

value : yield 표현식에서 반환된 값
done : 제너레이터 함수가 완료가 되었는지 나타내는 값

```
a.next()
1
▶ {value: 1, done: false}
```

```
a.next()
2
▶ {value: 2, done: false}
a.next()
3
4
▶ {value: 3, done: true}
```

```
a.next()
▶ {value: undefined, done: true}
```

제너레이터

```
> function* generator() {
      console.log(1);
    yield 1;
      console.log(2);
    yield 2;
      console.log(3);
      console.log(4);
    return 3;
  }
  const a = generator();
< undefined
> a.next()
  1
< ▶ {value: 1, done: false}
> a.return(5)
< ▶ {value: 5, done: true}
> a.next()
< ▶ {value: undefined, done: true}
```

```
function* generator() {
    console.log(1);
  yield 1;
    console.log(2);
  yield 2;
    console.log(3);
    console.log(4);
  return 3;
}
const a = generator();
undefined
try {
  console.log(a.next());  // { value: 1, done: false }
  console.log(a.throw("예외"));  // 예외 발생
  console.log(a.next());  // 이 부분은 실행되지 않음
} catch (error) {
  console.log("Error:", error);  // 예외 처리
}
1
▶ {value: 1, done: false}
Error: 예외
```