34장 이터러블

34.1 이터레이션 프로토콜

0. 머릿말

- 이터레이션 프로토콜이란? — ES6에서 도입된 순회 가능한 자료구조를 만들기 위해 ECMAScript 사양에 정의하여 미리 약속한 규칙
- ES6 이전 — 순회 가능한 자료구조(배열, 문자열, 유사 배열 객체, DOM 컬렉션 등)는 통일된 규약 없이 각자 나름의 구조를 가지고 for문, for...in문, forEach메서드등 다양한 방법으로 순회
- ES6 — 순회 가능한 자료구조를 이터레이션 프로토콜을 준수하는 이터러블로 통일하여 for ... of 문, 스프레드 문법, 배열 디스트럭처링 할당 대상으로 사용할 수 있도록 일원화
- 이터레이션 프로토콜 2가지
  - 이터러블 프로토콜 — Symbol.iterator를 프로퍼티 키로 사용한 메서드를 직접 구현하거나 프로토타입 체인을 통해 상속받은 Symbol.iterator 메서드를 호출하면 이터레이터 프로토콜을 준수한 이터레이터를 반환한다.
    - 이러한 규약을 이터러블 프로토콜이라고 하며, 이터러블 프로토콜을 준수한 객체를 이터러블이라고 한다.
    - 이터러블은 for .. of 문으로 순회할 수 있으며 스프레드 문법과 디스트럭처링 할당의 대상으로 사용할 수 있다.
  - 이터레이터 프로토콜 — 이터러블 Symbol.iterator 메서드를 호출하면 이터레이터 프로토콜을 준수한 이터레이터를 반환
    - next메서드 소유 — 이터레이터는 next 메서드를 소유하며 next 메서드를 호출하면 이터러블을 순회하며 value와 done 프로퍼티를 갖는 이터레이터 리절트 객체를 반환한다.
    - 포인터 역할 — 이러한 규약을 이터레이터 프로토콜이라고 하며, 이터레이터 프로토콜을 준수한 객체를 이터레이터라고 한다. 이터레이터는 이터러블의 요소를 탐색하기 위한 포인터 역할을 한다.

1. 이터러블

- 이터러블 프로토콜을 준수한 객체를 뜻함 — 즉, 이터러블은 Symbol.iterator를 프로퍼티 키로 사용한 메서드를 직접 구현하거나 프로토타입 체인을 통해 상속받은 객체를 말한다.
- 배열, 문자열, Map, Set 등은 이터러블
  - For of 문, 디스트럭처링, 스프레드 사용 가능
  - 일반 객체는 이터러블 X — 단, 2021년 1월부터 스프레드 문법 사용 허용

2. 이터레이터

- 이터러블의 Symbol.iterator 메서드를 호출하면 이터레이터 프로토콜을 준수한 이터레이터를 반환
- 이터러블의 Symbol.iterator 메서드가 반환한 이터레이터는 next 메서드를 갖는다.
  - next는 이터러블의 각 요소를 순회하기 위한 포인터의 역할
  - Next 메서드를 호출하면 순회 결과를 나타내는 이터레이터 리절트 객체를 반환
    - done은 순회 완료 여부
    - value는 현재 순회중인 이터러블의 값

34.2 빌트인 이터러블

- 해당 표준 빌트인 객체들은 빌트인 이터러블
  - Array — Array.prototype[Symbol.iterator]
  - String — String.prototype[Symbol.iterator]
  - Map — Map.prototype[Symbol.iterator]
  - Set — Set.prototype[Symbol.iterator]
  - TypedArray — TypedArray.prototype[Symbol.iterator]
  - arguments — arguments[Symbol.iterator]
  - DOM 컬렉션
    - NodeList.prototype[Symbol.iterator]
    - HTMLCollection.prototype[Symbol.iterator]

34.3 for ... of 문

- 이터러블을 순회하면서 이터러블의 요소를 변수에 할당 — for (변수선언문 of 이터러블) { ... }의 내부 동작 원리
  - 내부적으로 이터레이터의 next 메서드 호출로 이터러블을 순회하며 next 메서드가 반환한 이터레이터 리절트 객체의 value 프로퍼티 값을 for ... of 변수에 할당
  - 이터레이터 리졸트 객체의 done 프로퍼티 값이 false면 계속 순회 true 면 순회 중단
- for ... in문은 객체의 프로토타입 체인 상에 ○존재하는 모든 프로토타입의 프로퍼티 중에서 프로퍼티 어트리뷰트 [[Enumerable]]의 값이 true인 프로퍼티를 순회하며 열거

34.4 이터러블과 유사 배열 객체

- 유사 배열 객체란 — 마치 배열처럼 인덱스로 프로퍼티 값에 접근할 수 있고 length 프로퍼티를 갖는 객체
  - Length 프로퍼티를 갖기에 for문으로 순회 가능
  - 인덱스를 나타내는 숫자형식의 문자열을 프로퍼티 키로 가지므로 마치 배열처럼 인덱스로 프로퍼티 값에 접근
  - 예제
- 유사 배열 객체는 이터러블이 아닌 일반 객체 — 따라서 Symbol.iterator 메서드가 없기때문에 for ... of 로 순회 X
- arguments, NodeList, HTMLCollection, 배열은 유사 배열 객체이면서 이터러블
  - ES6에서 이터러블이 도입되면서 유사 배열 객체인 네 가지 객체에 Symbol.iterator 메서드를 구현했다.
  - 하지만 모든 유사 배열 객체가 이터러블인 것은 아니다. — Array.from 메서드를 사용하여 배열로 간단히 변환 가능
    - Array.from 메서드는 유사 배열 객체 또는 이터러블을 인수로 전달받아 배열로 변환하여 반환

34.5 이터레이션 프로토콜의 필요성

- 이터러블은 for ... of 문, 스프레드 문법, 배열 디스트럭처링 할당과 같은 데이터 소비자에 의해 사용되므로 데이터 공급자 역할을 한다.
  - 데이터 소비자 — for...of문, 스프레드 문법, 배열 디스트럭처링 할당
  - 데이터 공급자 — 이터러블 — Array, String, Map, Set, TypedArray, DOM 컬렉션, arguments
- 만약 다양한 공급자가 각자의 순환 방식을 갖는다면 데이터 소비자는 다양한 순회 방식을 지원해야 하므로 효율적이지 않다. — 이터레이션 프로토콜을 통해 데이터 소비자는 하나의 이터레이션 프로토콜만 지원하도록 구현하면 된다. — 효율성 증대
- 이터러블을 지원하는 데이터 소비자는 내부에서 Symbol.iterator 메서드를 호출하여 이터레이터를 생성하고 — 이터레이터의 next 메서드를 호출하여 이터러블을 순회하며 이터레이터 리절트 객체를 반환한다. — 그리고 이터레이터 리절트 객체의 value/done 프로퍼티 값을 취득한다.
- 이처럼 이터레이션 프로토콜은 데이터 소비자와 데이터 공급자를 효율적으로 연결하는 인터페이스 역할을 한다.

34.6 사용자 정의 이터러블

1. 사용자 정의 이터러블 구현

- 일반 객체를 이터레이션 프로토콜을 준수하도록 구현하는 것
- 예제

2. 이터러블을 생성하는 함수

- 앞서 살펴본 예젠느 내부 수열의 최대값 max를 가지고 있다. — 수열의 최대값을 외부에서 전달할 수 있도록 변경해보자
- 예제

3. 이터러블이면서 이터레이터인 객체를 생성하는 함수

- 앞서 살펴본 fibonacciFunc 함수는 이터러블을 반환한다. — 만약 이터레이터를 생성하려면 이터러블의 Symbol.iterator 메서드를 호출해야 한다.
- 예제

4. 무한 이터러블과 지연 평가

- 지연평가
  - 이터러블은 데이터 공급자 역할을 한다. — 배열이나 문자열 등은 모든 데이터를 메모리에 미리 확보한 다음 데이터를 공급한다. — 하지만 위 예제의 이터러블은 지연 평가(lazy evaluation)을 통해 데이터를 생성한다.
  - 지연평가는 데이터가 필요한 시점 이전까지는 미리 데이터를 생성하지 않다가 데이터가 필요한 시점이 되면 그때야 비로소 데이터를 생성하는 기법이다. — 불필요한 데이터를 미리 생성하지 않고 필요한 데이터를 필요한 순간에 생성
    - 빠른 속도
    - 불필요한 메모리 소비 X
    - 무한 표현 가능
- 무한 수열 구현
  - 예제
    - 예제에서 fibonacciFunc 함수는 무한 이터러블을 생성한다.
    - 하지만 데이터 소비자인 for of 문이나 배열 디스트럭처링 할당 등이 실행되기 이전까지 데이터를 생성하지 않는다. — 내부에서 next메서드를 호출할 때 데이터가 생긴다.